NASA Conference Publication 3187, Vol. I

Space Technology Interdependency Group

Sixth Annual Workshop on Space Operations Applications and Research (SOAR '92)

Volume I

Proceedings of a workshop held in
Houston, Texas
August 4–6, 1992

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1993	Conference Publication

4. TITLE AND SUBTITLE
Sixth Annual Workshop on Space Operations Applications and Research (SOAR '92)

5. FUNDING NUMBERS

6. AUTHOR(S)
Kumar Krishen, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lyndon B. Johnson Space Center
Houston, TX 77058

8. PERFORMING ORGANIZATION REPORT NUMBER
S-706

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546
U.S. Air Force, Washington, D.C. 23304

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CP 3187

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Publicly available
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Subject Category: 99

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document contains papers presented at the Space Operations, Applications and Research Symposium (SOAR) hosted by the U.S. Air Force (USAF) on August 4-6, 1992, and held at the Johnson Space Center (JSC) Gilruth Recreation Center. The symposium was cosponsored by the Air Force Materiel Command and by NASA/JSC. Key technical areas covered during the symposium were robotics and telepresence, automation and intelligent systems, human factors, life sciences, and space maintenance and servicing. The SOAR differed from most other conferences in that it was concerned with Government-sponsored research and development relevant to aerospace operations. More than 135 papers, 36 exhibits, 2 panel discussions, and several keynote speeches were included in SOAR '92. The USAF and NASA programmatic overviews were also held for each of the five technical areas. These proceedings, along with the comments and suggestions made by the panelists and keynote speakers, will be used in assessing the progress made in joint USAF/NASA projects and activities and to identify future collaborative/joint programs. The SOAR '92 symposium was the responsibility of the USAF/NASA Space Technology Interdependency Group (STIG) Operations Committee (SOC). Symposium proceedings include papers covering various disciplines presented by experts from NASA, the USAF, universities, and industry.

14. SUBJECT TERMS
Space, ground, operations, automation, robotics, life sciences, medical protocols, knowledge acquisition, expert systems, telepresence, virtual reality, training systems, biomedical, debris, space shuttle, Space Station Freedom, lunar outposts, Mars missions

15. NUMBER OF PAGES
740

16. PRICE CODE

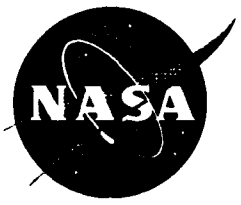
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

NASA Conference Publication 3187, Vol. I

Sixth Annual Workshop on Space Operations Applications and Research (SOAR '92)

Kumar Krishen, Editor
NASA Lyndon B. Johnson Space Center
Houston, Texas

Proceedings of a workshop sponsored by the National Aeronautics and Space Administration Washington, D.C., the U.S. Air Force, Washington, D.C., and cosponsored by the University of Houston-Clear Lake, Houston, Texas, and held in Houston, Texas August 4-6, 1992



National Aeronautics and Space Administration

Science and Technical Information Branch

CONTENTS

SECTION II: AUTOMATION AND INTELLIGENT SYSTEMS

Session A1: ADVANCED KNOWLEDGE-BASED SYSTEMS TECHNOLOGY
Session Chair: Dr. Abe Waksman

Session A2: PLANNING AND SCHEDULING I
Session Chair: Lt. Jennifer Skidmore

Session A3: PLANNING AND SCHEDULING II
Session Chair: Andrew Mayer

Session A4: MONITORING AND CONTROL
Session Chair: Dr. Robert Lea

INTRODUCTION

Kumar Krishen, Ph.D.

The Space Operations, Applications and Research (SOAR) Symposium and Exhibition is conducted by the Space Technology Interdependency Group (STIG) Operations Committee (SOC) on an annual basis. The goals of SOC are to (1) identify and characterize interdependent programs/projects, (2) encourage interdependent programs, (3) interchange technical and programmatic information and share lessons learned, and (4) identify critical voids and nonproductive overlaps in technology programs.

The reorganization of SOC was completed in the past year. This reorganization has resulted in five technical subcommittees on Robotics and Telepresence, Automation and Intelligent Systems, Human Factors, Life Sciences, and Space Maintenance and Servicing (fig. 1) being formed. More significantly, SOC has representatives from the U.S. Navy, Army, DARPA, SDIO, and the Department of Energy (DOE), in addition to NASA and the U.S. Air Force (USAF). The scope and membership of each subcommittee are given in figures 2 through 6. It is evident that both the depth and breadth of interdependency have been expanded and that the focus has been on efforts in research and technology that are related to both ground and space operations.

The SOAR Symposium and Exhibitions have been an excellent means to address most of the SOC goals. SOAR '92 brought together investigators and agencies to interchange technical information and share lessons learned. It was attended by more than 400 scientists and engineers. Many others visited the exhibits, which were free to the public. More than 135 papers, 36 exhibits, 2 panel discussions, and several keynote speeches were included in the program. The program organization committee is summarized in figure 7.

One of two welcome speeches for SOAR '92 was given by Dr. Billy E. Welch, the Director of USAF Armstrong Laboratory at Brooks Air Force Base, Texas. Dr. Welch noted there is much similarity between NASA and USAF objectives. Congress and higher levels of management are well aware of this similarity, and the two research programs need to be very closely coordinated. Scientists and engineers in both organizations need to share information by talking with other researchers involved in similar work, as well as with representatives from user organizations.

The NASA opening address was given by Dr. Robert Norwood, Deputy Director of Space Technology at NASA Headquarters. Dr. Norwood explained the evolution of the Agency integrated technology plan and the priorities that face the Agency. He pointed out the importance of operations research and technology, and the need for STIG to continue developing independency projects involving NASA, the Department of Defense (DOD), and DOE.

The keynote speaker for the banquet on August 5, 1992, was Dr. Allan Schell. Dr. Schell is the principal Assistant to the Deputy Chief of Staff, Science and Technology in the recently established Air Force Materiel Command. Dr. Schell discussed how the relationships between coordination efforts by STIG and those by the Joint Directors of Laboratories (JDL) are an important step in implementing a new coordination-oriented science and technology (S&T) management process.

Dr. Schell also discussed some of the important differences between the JDL and STIG. Specific JDL goals are to identify new opportunities for joint service S&T programs, plan and oversee execution of cooperative S&T programs, and coordinate the consolidation and collocation of laboratory functions. The primary function of the STIG is to improve technical data interchange by identifying candidate

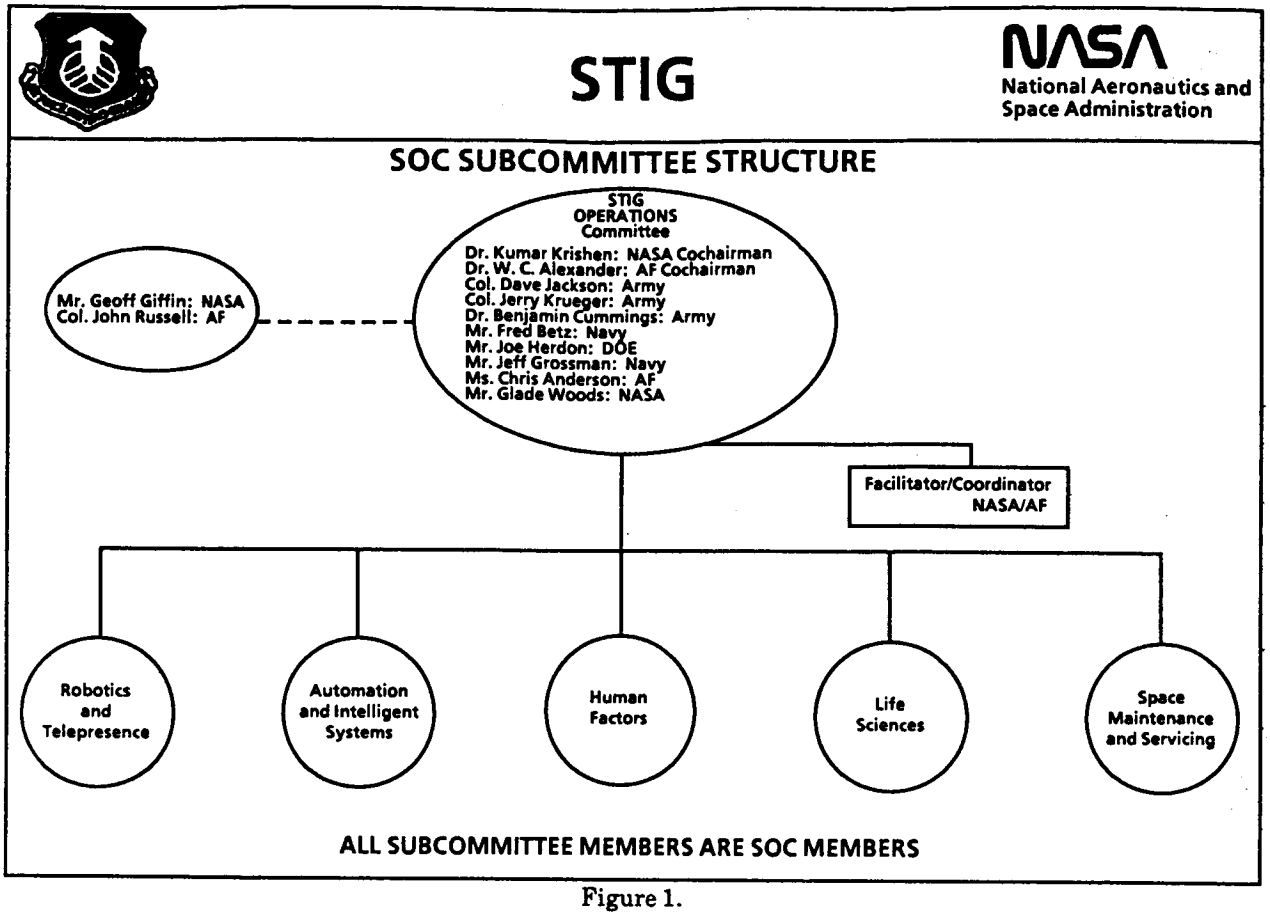


Figure 1.

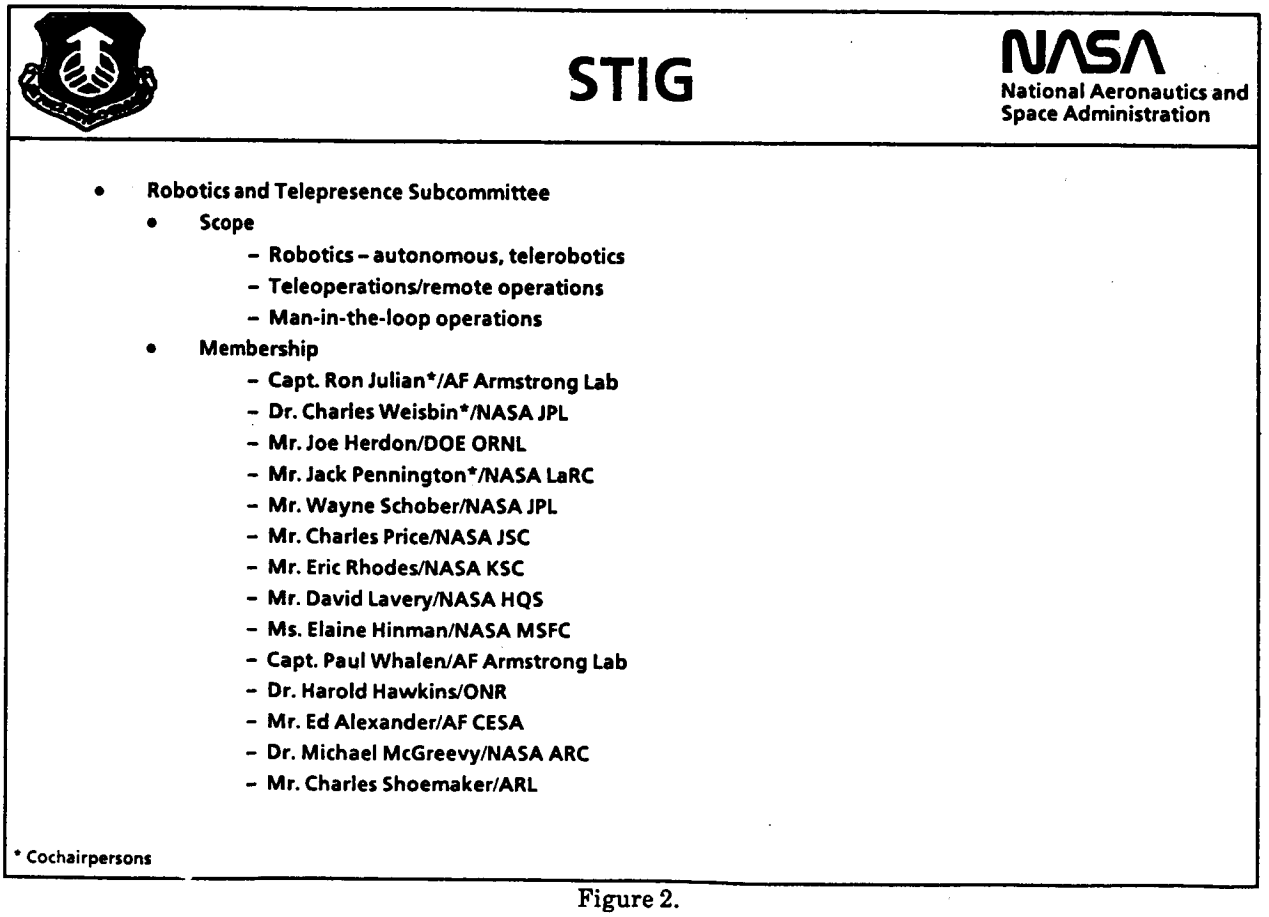


Figure 2.

STIG

NASA
National Aeronautics and Space Administration

- Automation and Intelligent Systems Subcommittee
  - Scope
    - Knowledge-Based Systems
    - Artificial Intelligence
    - Virtual Reality
    - Neural Networks
    - Fuzzy Logic
    - Vehicle Health Monitoring
  - Membership
    - Dr. Northrup Fowler, III*/AF Rome Lab
    - Dr. Peter Friedland*/NASA ARC
    - Mr. Robert Savely/NASA JSC
    - Ms. Nancy Sliwa/NASA KSC
    - Mr. Ralph Kissel/NASA MSFC
    - Dr. Melvin Montemerlo/NASA HQS
    - Dr. Abraham Waksman/AFOSR
    - Ms. Kathleen Healey/NASA JSC
    - Dr. Richard Doyle/NASA JPL
    - Mr. James Overholt/TACOM
    - Ms. Chris Anderson/AF Phillips Lab

* Cochairpersons

Figure 3.

STIG

NASA
National Aeronautics and Space Administration

- Human Factors Subcommittee
  - Scope
    - Human Performance – measurement and prediction
    - Extra- and intravehicular operations
    - Human-Machine interface
    - Training Systems
  - Membership
    - Col. Donald Spoon*/AF Armstrong Lab
    - Dr. Mary Connors*/NASA ARC
    - Ms. Tandi Bagian/NASA JSC
    - Dr. Jane Malin/NASA JSC
    - Mr. Stephen Hall/NASA MSFC
    - Mr. William B. Williams/NASA KSC
    - Dr. Kristin Bruno/NASA JPL
    - Dr. John Tangney/AFOSR/NL
    - Dr. Carl Englund/NRaD
    - Dr. Richard Monty/ARL/HRED
    - Dr. James Walrath/ARL/HRED
    - Dr. Jonathan Gluckman/Navy Air Warfare Center

* Cochairpersons

Figure 4.

STIG

NASA
National Aeronautics and Space Administration

- Life Sciences Subcommittee
  - Scope
    - Life Support
    - Health Systems
    - Biomedical research
    - Performance characterization
    - Medical operations
    - Crew Training Systems
    - Space Radiation Effects
  - Membership
    - Dr. Andrew Pilmanis*/AF Armstrong Lab
    - Dr. Gerald Taylor*/NASA JSC
    - Dr. Jerry Homick/NASA JSC
    - Col. Donald Spoon/AF Wright Lab
    - Dr. C. L. Snead, Jr./DOE
    - Dr. Gregory Nelson/NASA JPL
    - Capt. Terrell Scoggins/AF Armstrong Lab
    - Lt. Col. Roger U. Bisson/AF Armstrong Lab
    - Dr. Phil Whitley/Navy Air Warfare Center
    - Col. Jerry Krueger/USA RIEM

* Cochairpersons

Figure 5.

STIG

NASA
National Aeronautics and Space Administration

- Space Maintenance and Servicing
  - Scope
    - Maintenance and repair operations
    - Assembly operations
    - Servicing operations
    - Fault detection
    - Nondestructive Evaluation
  - Membership
    - Lt. Col. Gary Johnson*/AF Phillips Lab
    - Mr. Chuck Woolley*/NASA JSC
    - Mr. E. C. Smith/NASA MSFC
    - Col. George Sawaya/USSPACECOM
    - Mr. John Cox/USAF SSD
    - Mr. Bill Eggleston/NASA JSC
    - Mr. Jeffrey Hein/NASA JSC
    - Dr. Neville Marzwell/NASA JPL

* Cochairpersons

Figure 6.

SOAR '92 will include USAF and NASA programmatic overviews, panel sessions, exhibits, and technical papers in the following areas:

- Robotics and Telepresence
- Automation and Intelligent Systems
- Space Maintenance and Servicing
- Human Factors
- Life Support

Exhibit Hours

Tuesday, August 4	10:00 am – 7:00 pm
Wednesday, August 5	8:00 am – 7:00 pm
Thursday, August 6	8:00 am – Noon

Welcome/Opening Addresses (August 4, 8:30 am – 9:30 am)

Dr. Billy Welch, Director of Armstrong Laboratory
Dr. Robert Norwood, Deputy Director for Space Technology, NASA/OAST

Panel Discussion (August 4, 3:30 pm – 5:00 pm)

Technology Implementation and Transition

Moderator:	Dr. Kumar Krishen
Panelists:	Mr. Robert Savely Mr. Mark Gersh Mr. Wayne Schober Mr. Steve Riemer Dr. Jeff Kantor Mr. Melvin Rogers

Keynote Dinner Session (August 5, 7:00 pm – 9:00 pm)

Welcome and Opening Remarks	Mr. Geoff Giffin, Deputy Director Operations Thrust NASA/OAST
Keynote Speakers:	Dr. Allan Schell, Principal Assistant, DCS/Science and Technology Air Force Materiel Command

Exhibitors

Armstrong Lab, USAF
Analysis & Technology
Computer Sciences Corp.
Cybernet Systems Corp.
GE Government Services
GHG Corporation
Intermetrics
Kinesix
Krug Life Sciences
LinCom Corp.
Lockheed ESC
Loral Space Information Systems
McDonnell Douglas
Merit Technology, Inc.
NASA/Johnson Space Center
Oracle Corp.
Programming Research Corp.
Rice University
Silicon Graphics, Inc.
Space Industries
Togai InfraLogic
Virtual Prototype
XYPlex, Inc.

Symposium Coordinators

Symposium Cochairs:	• Dr. Kumar Krishen NASA/JSC
	• Dr. W. C. Alexander AL/XP
Technical Coordinators:	• Mr. Robert Savely NASA/JSC
	• Co. John Tedor AL/XPT
Administrative Cochairs:	• Ms. Carla Armstrong Barrios Technology, Inc.
	• Mr. Dick Rogers Lockheed/ESC
	• Ms. Lana Arnold Lockheed/ESC
	• Dr. Glenn Freedman University of Houston–Clear Lake
Exhibit Cochairs:	• Mr. Chris Ortiz NASA/JSC
	• Mr. Ellis Henry I-NET, Inc.
	• Dr. Don Myers University of Houston–Clear Lake

Technical Area Coordinators

	USAF	NASA
Robotics and Telepresence	Capt. Ron Julian AL/CFBA (513) 255-3671	Dr. Charles Weisbin NASA JPL (818) 354-2013
Automation and Intelligent Systems	Dr. Northrup Fowler III RL/C3C (315) 330-3011	Dr. Peter Friedland NASA ARC (415) 604-4277
Human Factors	Col. Donald Spoon AL/CF (513) 255-5227	Dr. Mary Connors NASA ARC (415) 604-6114
Life Support	Dr. Andrew Pilmanis AL/CFTS (512) 536-3545	Dr. Howard Schneider NASA JSC (713) 483-2380
Space Maintenance and Servicing	Lt. Col. Gary Johnson PL/XP-A (505(846-9735	Mr. Charles Woolley NASA JSC (713) 283-5362

Figure 7.

interdependent programs, encouraging new cooperative relationships, and sharing the results of cooperative and interdependent S&T. The JDL activities and the STIG activities are highly complementary, and both are needed to get the most out of the space S&T budget.

A second speech at the banquet was given by Mr. Geoff Giffin, Deputy Director, Operations Thrust, NASA Office of Aeronautics and Space Technology. Mr. Giffin described the evolution of the operation thrust plans in the Agency. He then related the importance of SOAR '92 to the overall implementation of the operations thrust.

The SOAR symposia (SOAR '92 is the sixth) have been very successful at sharing information and improving coordination. Future plans call for greater participation by the Army, Navy, and DOE representatives—in addition to NASA and Air Force representatives. Opportunities will be provided for mutual goal setting, information sharing, and cooperative long-range planning. As a result of these efforts, it is believed that Operations Research and Development (R&D) will set new standards for quality and relevance—standards that will benefit the Nation.

WELCOME/OPENING ADDRESSES

Dr. Billy Welch
Director, Armstrong Laboratory

Welcome to the Sixth Annual Workshop on Space Operations, Applications, and Research (SOAR). The conference is cosponsored by NASA/Johnson Space Center (JSC) and the Air Force Materiel Command. As in any joint effort, a high degree of teamwork on the part of personnel from both organizations is required to put on each conference. Since the conference is being held at JSC this year, NASA did a lot of the work in getting things organized. Special thanks are due to Dr. Kumar Krishen and Mr. Bob Savely of JSC as well as to key personnel in the NASA subcontractor organizations that helped set up and manage the conference.

The papers that will be presented look very interesting and exciting. I plan to attend many of these sessions myself, and I hope that you do, too.

SOAR has slightly different objectives from most other conferences. One of the major objectives of SOAR '92 is to coordinate plans for future research and development (R&D). Increased coordination is need to (1) avoid duplication of effort, (2) make full use of information, and (3) plan joint efforts. This kind of coordination is mandatory in the budget-crunching world of today. The U.S. Congress, the President, NASA management, the Department of Defense, and the U.S. Air Force (USAF) want to be certain that the Government does not fund the same R&D programs in two different organizations at the same time. In accordance with these policies, the SOAR symposium has been expanded. In addition to representatives of NASA and the USAF, the conference now includes representatives of the Department of Energy, the U.S. Army, and the U.S. Navy. We expect each of these organizations to play a very active and important role in SOAR '93.

The amount of interest in SOAR varies from one Government agency to the next, but all of these organizations have important R&D areas that need to be coordinated.

The SOAR '92 conference differs from other conferences because it is primarily concerned with Government-sponsored R&D as well as with aerospace operations and joint efforts between R&D and aerospace operations. SOAR differs from the usual R&D conferences, however, because it involves discussions about plans for future R&D as well as reports on completed R&D efforts.

USAF interests in space-related technology are best described by the mission areas of the four Super Labs—Armstrong Laboratory (AL), Phillips Laboratory, Rome Laboratory, and Wright Laboratory. As is shown in the rather complicated slide for the Armstrong Laboratory, AL work covers a lot of territory. Anywhere people are involved, AL is involved. The second slide illustrates the work—a lightweight, exo-atmospheric projectile—performed at Phillips Laboratory. The third slide shows the work that Rome Lab is doing in exploring the technological basis of space-based radar systems. And the fourth and last slide in this series illustrates work being conducted by Wright Laboratory on computational fluid dynamics.

Because of the differences in laboratory missions, the four Super Labs are interested in different areas of space-related R&D, as shown in the following table.

	Armstrong	Phillips	Rome	Wright
Robotics & Telepresence	X			X
Automation & Intelligent Systems	X	X	X	X
Human Factors	X		X	X
Life Support	X			
Space Maintenance	X	X		

The SOAR conference will focus on many concrete USAF and NASA plans for the future. I think you will find that the two sets of plans—those of the USAF and those of NASA—are quite similar. For example, NASA has two sets of plans for man in space, both as a physical presence and as a robotic telepresence. Similarly, the USAF, the U.S. Army, and the U.S. Navy have two sets of plans for man in combat: a physical presence and a robotic telepresence. Three photos illustrate these plans: (1) the first is of robotics telepresence work being conducted by NASA, (2) the second shows two robot arms holding screwdrivers, and (3) the third is a photo of some robotics work being conducted at AL. If these photos look pretty much the same, you are right—and that is precisely the point I am trying to make.

The same kind of similarity can be found when addressing life support and maintenance problems. NASA experiences life support problems at super high altitude and the USAF encounters life support problems at high altitudes. Similarly, NASA has maintenance problems in space while the USAF faces similar maintenance problems on the ground. To demonstrate these problems, three slides are shown depicting (1) an ejection capsule for high-altitude aircraft; (2) important adjustment problems in space; and (3) someone at Phillips Lab making adjustments to an exo-atmospheric projectile on the ground.

In addition to pointing out similar experiences among organizations, the SOAR '92 conference also outlines some long-range hopes for the future. Among these hopes are a triple capacity for job aides and information collection devices; new systems that will make manned flight routine, casual, and inexpensive; and first-generation "holodecks." The portable computer that appears in the first slide provides all of the information currently contained in the many Technical Manuals also shown in the photograph. The second slide is of the Space Shuttle. Everyone, of course, has high hopes for the next generation Space Shuttle. But, the USAF is working on plans for a National Aerospace Plane (NASP), which has different objectives from those of the Space Shuttle. Obviously, plans for these two aerospace vehicles—the Space Shuttle and the NASP—need to be closely coordinated. The next slide presents a version of the AL "supercockpit", which uses virtual imagery to project an image of the world outside of the high-speed aircraft. The pilot can either be in a completely enclosed cockpit or on the ground.

I know that you will benefit from the information exchange at the SOAR '92 conference, but I hope you will also share your dreams for the future. My own dreams for the future include multisystem virtual environments for maintenance training, an ability to reach and strengthen enhancement tools for use in space, super-intelligent robots for exploring distant planets, easier "in-orbit" spacecraft maintenance, high-fidelity telepresence equipment, and lightweight, high-strength, super-flexible life support suits.

Whatever happens in the future, let yourself dream a little during the proceedings. I assure you that if we see anyone daydreaming during any one of our lectures, we will assume that the daydreams are

technical. We also will assume that they are concerned with important developments in the distant future rather than with the conditions that exist today.

But be assured that you and your ideas are both welcome here—whatever your ideas or daydreams are. We hope that you, in turn, will pick up some new ideas from your fellow professionals, and that any ideas gathered from this conference will create new visions for you regarding where we should be going with aerospace-related R&D. Thank you.

Dr. Robert Norwood
Deputy Director for Space Technology, NASA/OAST

The vision for space research and technology research and technology (R&T) is "World leadership in space research and technology development to make it possible to look beyond the known, to challenge the limits of human capability, to inspire the generation of the 21st century, and to secure the benefits of space for life on Earth. The OAST R&T mission is to "provide technology for future civil space missions and [to] provide a base of research and technology capabilities to serve all national space goals." To meet these ends, 20-year strategic visions have been stipulated for the following:

- Space Science—Technologies will be ready to enable low mass, facility-class single aperture and interferometric space-based observatories across the EM spectrum; to conduct cost-effective, long-term remote sensing to make complex, but frequent, *in situ* scientific studies in space laboratories on the Moon and at the planets; and to enhance human understanding of extremely large science data sets.

- Planetary Surface—Technology will be completed to emplace safe and permanent, largely self-sufficient human outposts on the Moon or Mars, with capabilities for extensive surface exploration and science and for resource exploitation operations.

- Transportation—Capabilities will be in hand to enable safe, highly operable reusable piloted vehicles for ETO transport; low cost, reliable expendable ETO vehicles for small, medium, and large payloads (including internationally competitive ELVs); and long-life, high-performance space transfer systems that enable human exploration of the Moon and Mars and also ambitious deep space robotic missions.

- Space Platforms—Technologies will be ready for long-lived Earth orbiting platforms with significantly reduced masses and costs but increased payload capabilities (both manned and unmanned), and for reduced mass, high-reliability spacecraft for long-duration deep space science and exploration mission applications.

- Operations—New technology will make possible largely autonomous ground, flight, and in-space systems that will reduce the costs of civil space mission operations and of the infrastructure while improving their safety and reliability, thus enabling more complex capabilities and massively increasing mission data returns.

- Innovative Discipline Research—Innovative, high-leverage concepts will be validated both analytically and in laboratory research—concepts that make possible "next generation" Earth and space science and exploration and commercial and infrastructure missions.

The way in which these various technologies will be integrated for the civil space program are shown in figure 1. Three steps will lead to a successful integrated technology plan process: (1) solicit internal and external needs through mission forecasts and an in-depth review of technology reviews and priorities; (2) develop an integrated technology plan by forming teams with technologists and users and by establishing decision rules for base and focused programs; and (3) conduct an external review.

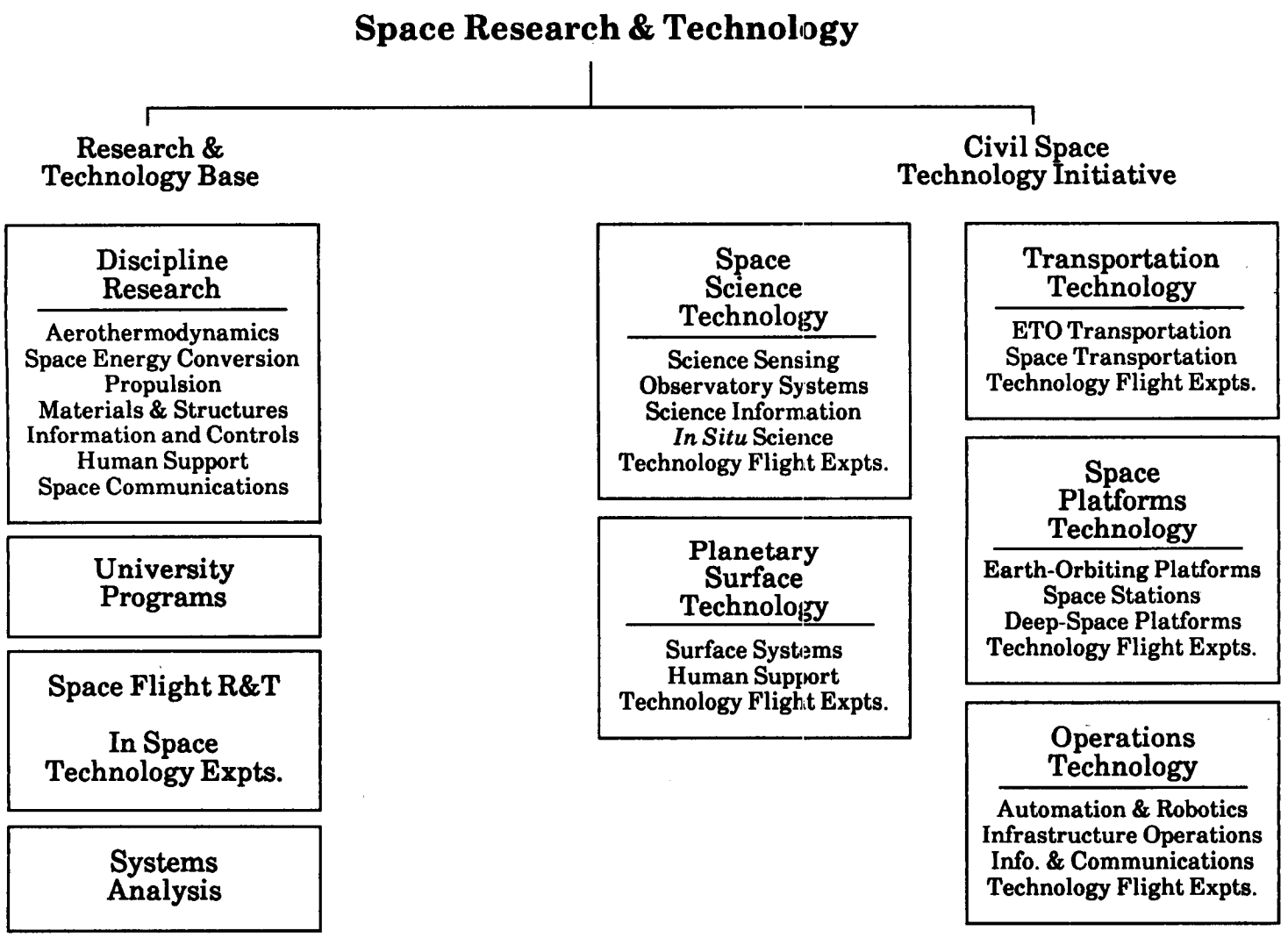


Figure 1. Integrated technology plan for the civil space program.

Various flight programs are forecast at the completion of the proposed R&T strategy. These are as follows:

- A 5-year forecast for 1993 through 1997 with limited starts, which includes: completion of initial Space Station Freedom, some Shuttle improvements, initial EOS and EOSDIS, selected space science starts, NLS development, initial SEI architecture selection, evolving GEO commercial COMMSATs, and minor upgrades of commercial ELVs.

- A 10-year forecast for 1998 through 2003 with multiple new starts to be launched in 2003 through 2010, which includes SSF evolution/infrastructure, final Shuttle enhancements, advanced LEO EOS platforms/full EOSDIS, multiple space science starts, NLS operations/evolution, evolving launch/operations facilities, initial SEI/lunar outpost start, DSN evolution (Ka-band communications), new GEO commercial COMMSATs, and new commercial ELVs.

- A 20-year forecast for 2004 through 2011 includes multiple operations for new starts to be launched in 2009 through 2020, which includes SSF-Mars evolution, beginning of AMLS/PLS development, multiple space science starts, DSN evolution (optical COMM), initial Mars HLLV development, evolving lunar systems, Mars SEI architecture chosen, large GEO COMMSATs, and new commercial ELVs.

To fulfill these forecast needs, external technology sources will be tapped. Among these will be the following:

- Boeing Aerospace & Electronics
- Gencorp-Aerojet
- General Electric–Philadelphia
- General Electric–Valley Forge
- Grumman
- Hughes
- Martin Marietta
- McDonnell Douglas
- RCA
- Space Systems/Loral
- Sparta
- Stanford Telecom
- TRW
- United Technologies Corporation

Additional needs will be supplied through direct inputs from SSTAC/ARTS members and earlier NRC survey data.

Once the external technology sources are fully tapped, operations technology can be moved forward. The goal of operations technology is to develop and demonstrate technologies to reduce the cost of NASA operations, improve the safety and reliability of those operations, and enable new, more complex activities to be undertaken. This operations thrust supports the following major activities: (1) in-space operations, (2) flight support operations, (3) ground servicing and processing, (4) planetary surface operations, and (5) commercial communications. The following technology areas are included: (1) automation and robotics, (2) infrastructure operations, (3) information and communications, and (4) flight experiments. While undertaking this operations thrust support, the following 10-year objectives will be met:

- Infrastructure Operations—Demonstrate workstation-based command generation technology to reduce mission flight control operations costs by 75%. Also, complete R&T for real-time spacecraft operations analysis tools to reduce spacecraft design, development, test, and integration costs by 40%.

- Automation and Robotics—Demonstrate capabilities using advanced manipulators, sensors, exoskeletons, and controls to enable 25% of all scheduled and contingency EVA tasks to be performed by telerobotics systems. Also, provide technology for ground- and space-based artificial intelligence systems to perform 40% to 60% of spacecraft fault detection, identification, and isolation.

- Information and Communications—Demonstrate space data systems and communications link technology to provide 100:1 improvement in end-to-end information and data management system performance (while costs are reduced by 75%) complete flight demonstration of very high data rate optical communications systems.

Operations technology may be broken down into the following components:

- Automation and Robotics—Including mission control support, planning and scheduling, ground servicing and support roles, and in-space teleoperation and telerobotics.

- Infrastructure Operations—Including in-space assembly and construction, space processing and servicing, training and human factors, ground test and processing, and flight control and space operations.

- Information and Communications—Including space data systems, ground data systems, commercial satellite communications, photonics systems, and high-rate communications.

- Flight Experiments–Including commercial satellite communications and optical communications.

For the 10-year space technology plan, certain critical elements of NSPD-4 will lead to a space launch strategy, which is detailed as follows:

- Ensuring space launch capabilities meet needs.

- Developing a new space launch system—unmanned, but man-ratable; reduce operating costs; and improve reliability, responsiveness, and performance.

- Sustaining vigorous space launch technology program—provide cost-effective improvements to current launch systems; and support development of advanced launch capabilities complementary to the new launch system.

- Actively considering commercial space launch needs.

NSPD-4 strategy guidelines will stipulate the need for space launch technology, as follows:

- Long-term, broadly based research and focus technology programs to support national space launch capabilities—launch system components, among which are engines, materials, structures, and avionics; upper stages; improved launch processing concepts; advanced launch system concepts, including SSTO (which includes NASP); and experimental flight vehicle programs.

- Ten-year space launch technology plan—DOD, NASA, and DOE coordinated.

Current and future options in the technology planning areas are systems analysis and design; propulsion; structures, materials, and manufacturing; avionics; aerothermodynamics and recovery; and operations and processing.

The following summarizes requirements for the 10-year space technology plan:

- Improve launch reliability—Launch reliability should be 98% or better for unmanned cargo vehicles and 99.6% or better for manned vehicles.
- Reduce the operations and maintenance costs toward a goal of $1000 per pound to LEO.
- Achieve operational flexibility by removing most (or all) of the constraints to launching on schedule.
- Provide larger margins for new mission capabilities.
- Make launch vehicles and infrastructures environmentally compatible.

Future program options for operations and processing are as follows:

- Test facilities and instrumentation—Among which are fiber optic vehicles nets; automated bit/ go-no go; and NDE/automated ground handling.
- Propellant systems—Inexpensive, reliable cryogenic leak detection.
- Advanced management tools—Among which are streamlined mission planning and simplified/automated documentation process.

The challenge for the future is simple. We must improve launch reliability, significantly improve ground handling speed and flexibility and incur lower cost, and automate in-space operations.

KEYNOTE ADDRESSES

Mr. Geoff Giffin
Deputy Director, Operations Thrust NASA/OAST
Keynote Dinner—Welcome and Opening Remarks

Ladies and gentlemen, good evening. On behalf of "Pete" Petersen and the Office of Aeronautics and Space Technology (OAST) I want to welcome all of you to this SOAR symposium. I am very honored to be here again and want to thank Dr. Alexander and Dr. Krishen for the excellent job they have done in putting on this conference. Also, I would like especially to recognize the support staff who make it all possible. The speakers, panelists, and exhibitors are to be commended for the high quality and relevance of their contributions to making this symposium a success.

The thrust of this conference, as you all know, is *operations*. To my knowledge, this is the only symposium that takes place on a regular basis that has this focus. Operations has always been an important aspect of our jobs, and it is appropriate that it should be the subject of a forum such as this. Interestingly, it seems that the rest of the world is now catching up with us and putting more and more emphasis on the cost of operations and on the technologies that may be used to mitigate them. For the first time in recent memory, operations has been explicitly identified as an area needing urgent attention. Within NASA, the mission offices are citing the need for improved related technologies in addition to the more traditional calls for better sensors, bigger and faster rockets, and enhanced performance systems.

So what does this mean to us? It is clearly a challenge. We live in an era of flat or shrinking budgets. The days of constant growth in NASA and defense budgets appear to be behind us. If we want to do new things, we must reduce the cost of doing existing things—that means operations. Perhaps our slogan should be that "It is easier to get a dollar out of operations than it is to get a dollar out of the Congress." We need continually to look for better ways to do things and to press for opportunities to implement them and to look outside our normal world for models and inspiration. While we are busy working to develop these new technologies and field them, I think it is critical that we step back and examine our perspective. One thing I like to do when trying to divine the future is to project backwards as far as I'm trying to go forward and look at what has happened since that time. For example, who could have predicted a mere 15 years ago today's world of office automation and microcomputers?

You have heard often that we live in historic times—probably without anyone explaining what that means. Living through historic times is disconcerting, frustrating, uncertain, and full of fear. It's only the historians who look back and make sense of it all—perhaps that's why they are called historic times—partly because they know how it comes out. I have had some interesting conversations who my mother, who was a newlywed in England at the beginning of the Second World War, over the last couple of years. She vividly remembers those stirring speeches of Winston Churchill that sound so lifting and inspirational to us. At the time, they scared the pants off everyone because they made it sound as if invasion and defeat were imminent—which, of course, they were. The big difference is that we know how it came out!

We are obviously living through a period of change that is unequaled over probably the last 500 years, and it behooves us to examine our activities in light of that change. The demise of the "enemy," the Soviet Union, is not the end but, rather, the beginning of changes that will affect us all over the decades to come.

In the early days of civilization, we moved from tribal or village societies to city-states, and these eventually evolved into nation-states that were defined by two imperatives—the need to control a large enough area in which to conduct trade and commerce effectively, and the need for a viable offensive and defensive military capability. Ethnic diversity was generally subjugated to the "national will." This has been the model of political entities for about half a millennium; but it now appears to be evolving into the post-nation-state organization that might be called the "ethnic enclave state." This is made possible by two developments—(1) the mise of the multinational cooperation as the normal vehicle of trade and (2) the combined effect of changes in the nature of strategic warfare with the rise of a truly multinational or supranational military systems—e.g., NATO, the United Nations, etc.

We are simultaneously witnessing some of the joys of this process (the Baltic states) as well as some of its horrors (as expressed is the dissolution of Yugoslavia). But, whether we like it or not or approve of it or not, it will continue.

This is all very interesting (or maybe not after such a long day), but what does it mean to us? It means, I think, that we must not anticipate that the future will be a logical extension of the past. We are, I believe, in the midst of changes unprecedented in our lifetimes and, perhaps, in centuries. Not since the defeat of the Spanish Armada by the Combination of Sir Francis Drake and some very nasty weather in 1588 has a world power been eclipsed as drastically as the Soviet Union has been. We are part, and an important part, of the response that the United States will make to this strange and bewildering opportunity. Let me make sure that we take full advantage of it by pressing our view of the future and going forward.

As Robin Williams has said, both as the leader of the Dead Poet's Society and Peter Pan, *Carpe Diem*. Seize the day; make a difference. Thank you very much.

Dr. Allan Schell, Principal Assistant,
DCS/Science and Technology,
Air Force Material Command

Ladies and Gentlemen, it is an honor for me to be with you this evening. On behalf of Brigadier General Richard Paul, the Air Force Materiel Command Deputy Chief of Staff for Science and Technology, and myself, we both wish to extend our appreciation to this symposium's organizing committee for the invitation to speak to you tonight. In light of the major changes that have taken place within the Air Force since we gathered last year, General Paul and I are pleased with the opportunity to convey the continuing and, in many regards, growing importance that this Space Technology Interdependency Group (STIG) plays in Air Force space research and development.

Last year at this symposium, Major General Robert Rankine stated the importance of space to our national defense and to sustaining our technological and industrial strengths. A year later, the importance of space has certainly not diminished. Indeed, as we move out from under the Soviet-centered threat that has largely driven our national defense perspective for the nearly 4 decades of the space age, we are now starting to gain a better understanding of the possibilities, benefits, and excitement that space, both as a distinct natural resource and as a growing sphere of operations, can bring to the Air Force. One clear indication of the importance of space to the Air Force is its inclusion in our vision—the defining statement that guides all Air Force policy, missions, and organizational objectives.

General Ronald W. Yates, Commander of the Air Force Materiel Command, the new Air Force MAJCOM charged with cradle-to-grave acquisition and support for Air Force systems, has emphasized the need to integrate space fully into all our command functions. If the old saying that "everything in its time" is true, then we are certainly entering the era of space. The answer to any lingering questions about the importance to space to the Air Force is quite unmistakable—it is important, it is here to stay, and it will increasingly become an integral part of Air Force missions and operations.

The primary reason that we are at this conference is that NASA and the Department of Defense are the two principal national organizations charged with maintaining and enhancing our national capabilities in space. Over the last 40 years, we have been close partners in opening the space frontier. Our tremendous successes in space, in both crewed and robotic activities in civilian and defense areas, are the direct result of this close partnership and your hard work. In these times of changing national priorities, evolving organizational missions and responsibilities, and, perhaps most importantly, an expanding national consensus on the value of our space capabilities, we need to ensure that our partnership remains strong and effective in achieving our nation's space goals.

The STIG is the mechanism that we use to strengthen our successful partnership. Through the STIG, NASA and the Air Force have been successful in coordinating our activities to maximize the benefits of our space development programs. This success is recognized throughout the government. Recently, the Army, the Navy, and the DOE have joined NASA and the Air Force as STIG members. NOAA is in the process of joining, and we are looking to the future when DARPA and SDIO may wish to join. With such an expansion of the STIG, I am confident space technology development throughout the government will benefit.

The challenge we face with a growing number of organizations participating in the STIG is to maintain its vitality. The STIG is organized to draw upon the knowledge and experience of our very talented scientists and engineers to help us, corporately, decide how best to maximize our space development activities. It is a decentralized working group, built upon the concept that the working scientists and engineers best know their capabilities and qualities. We have learned through our Total Quality experience that engaged and motivated team members are the best way to succeed in achieving important goals such as our national goals in space. This SOAR Conference is a model process for achieving effective coordination of our R&D activities and sharing of the resulting technical data. As the STIG expands, let us make sure that we maintain this vitality and do not lose sight of the STIG primary objectives of technical data sharing and project coordination.

As we are all certainly aware, the major changes in the world are being reflected in changes with the armed services and, of particular interest to many attending this conference, in the Service laboratories. In response to a call by OSD, the Service laboratories have created a structure for science and technology coordination and consolidation. Referred to as Tri-Service S&T Reliance, it was charted to: enhance service science and technology, ensure that a critical mass of resources remains available to develop "world-class" products' reduce redundant capabilities and eliminate unwarranted duplication; gain S&T efficiency through collocation and consolidation of in-house work, when appropriate; and preserve the Service's mission-essential capabilities as the reductions, consolidations, and collocations are undertaken. The objective of Reliance is to identify effective ways to move the Service laboratories from primarily a framework of inter-service coordination to a framework of inter-service reliance, as the project name implies.

By November of last year, all three Services had directed the implementation of a new joint-service S&T management and planning process that would implement the recommendations of the Reliance project within the structure of the Joint Directors of Laboratories (JDL). Reliance and the resulting JDL are necessary and important steps to ensure that our S&T programs continue to meet our national defense technology needs in these times of diminishing defense budgets.

The JDL panel structure currently has 13 technical panels and one management panel. Within each panel, there are a number of subpanels—such as the Space Vehicles Panel. The initial success of the JDL in achieving the Project Reliance objectives has led to interest in expending participation in the JDL to include DARPA and SDIO. With a substantial portion of DARPA and SDIO technology development work being done by or through service laboratories, such a formal planning and coordination relationship should be beneficial to enhancing our overall S&T planning function.

We are now about a year into the implementation of the JDL process. Some of our early accomplishments of interest to this group include

- Collocation of in-house S&T work addressing Space-Based Wide-Area Surveillance Radar at the Air Force Rome Laboratory, Rome, New York

- Collocation of in-house S&T work addressing Space-Based Infrared Sensors for Wide-Area Surveillance at the Naval Research Laboratory, Washington, D.C.

- Collocation of all Biodynamics S&T of the Army and Navy to Armstrong Laboratory at Wright-Patterson Air Force Base, Dayton, Ohio

As naturally would be expected with such major changes in long-established organizational structures, there have certainly been a few bumps in the process. But, overall, the changes are being implemented relatively smoothly, principally because all participants recognize the importance of making this transition a success.

An advantage of organizing the Services' S&T functions under the JDL panel structure is that it improves our understanding of the interactions between the JDL technical panels and the STIG technical committees. One of the essential elements of effective technology sharing is just knowing who to call to ask questions or to make a point of meeting, at joint conferences such as this, to establish productive working relationships. Now, via the JDL and the STIG, researchers throughout the Service laboratories, NASA, and our partner government agencies have a structure for establishing and improving these working relationships.

One point that I would like to make is that there are distinct differences in the functions of the JDL and the STIG. Whereas both the JDL and the STIG address technology sharing, the JDL emphasizes the programmatic aspects and the STIG emphasizes the technical aspects. Specific JDL goals are to: identify new opportunities for service S&T programs; plan and oversee execution of cooperative S&T programs; and coordinate the consolidation and collocation of laboratory functions, where appropriate. The primary function of the STIG is to improve technical data interchange by: identifying candidate interdependent programs; encouraging new cooperative relationships; and sharing the results of a cooperative and independent S&T. The JDL activities and the STIG activities are highly complimentary and very important to getting the most out of our space S&T budget.

As I mentioned, the Air Force is beginning to see space in a new light. Desert Shield and Desert Storm demonstrated the value of space systems to our fighting forces. Space systems played a major part in helping achieve both our military and our political goals during the conduct of these operations. The fallout of this experience in terms of our space S&T has been very beneficial. Space, as I highlighted, is clearly integrated into the Air Force vision. But, the fallout goes farther than this. The Air Force is starting to look towards our space S&T in terms of achieving national space goals and thinking beyond purely Air Force or military systems. This is one of the benefits of the demise of the cold war and the changing perspectives that are gaining acceptance.

The Air Force currently has under way a number of R&D programs at Phillips and Rome Laboratories that could directly support space science, space commerce, and the President's Space Exploration Initiative (SEI). Let me highlight a few of these efforts that will be of value beyond military applications.

ELITE, which stands for Electric Insertion Transfer Experiment, is one of our Advanced Technology Transition Demonstrations, or ATTDs. It will flight demonstrate all of the specific technologies required for an electric-powered orbit transfer capability. Advanced electric propulsion uses solar electric or space nuclear electric power to power a high-performance arcjet rocket using liquid hydrogen. The potential benefit of this capability to the civilian space program is reduced launch costs through the utilization of smaller launch vehicles, and the ability to launch larger satellites on the same launch vehicle or to boost interplanetary spacecraft to higher velocities and reduce the transit time to reach other planets. If the current schedule holds, ELITE will be ready for flight demonstration in FY96.

TAOS, or Technology for Autonomous Operational Survivability, is another flight demonstration ATTD. It focuses on demonstrating decreased satellite ground support requirements through autonomous satellite operability and demonstrating space-qualified standard satellite components, such as the data bus, GPS receivers, and spaceborne computer. These technologies would be applicable to virtually all space systems, thus providing decreased development time and cost, lengthened operational lives, and reduced operating costs by minimizing ground support requirements. TAOS is planned for launch next year on the first DARPA TAURUS Standard Small Launch Vehicle.

ASCM, or Advanced Spaceborne Computer Module, is a standardized, producible, radiation-hardened computer module based on the standardized 1750A architecture. This system will offer an order of

magnitude improvement in processing capability. It is currently baselined by 32 DOD and NASA systems with an expected savings of $10M per satellite. Related development efforts have already generated radiation-hardened 64 Kbit and 256 Kbit static memory chips that are seeing application in several programs, including use by NASA.

Environmentally acceptable propellant technology development involves two areas important to our continued use of solid rocket boosters. One development focuses on environmentally acceptable ways of recycling existing solid rocket booster propellants to allow reuse of this resource in newly fabricated launch vehicles. We are looking into ways of extracting the propellants from decommissioned ballistic missiles and reusing the propellants in other civilian and military space launch systems. Another related technology is the development of environmentally acceptable solid propellants that minimize environmentally damaging combustion byproducts such as hydrochloric acid. Together, these two programs will enhance the acceptability of continued use of solid rocket boosters at a time of growing environmental awareness.

One of the most impressive areas of Air Force research is advanced optics using adaptive optics and guide-star technologies to significantly boost the performance of ground-based observatories. Work in this area has turned the ground-based observation world upside down by returning ground observatories to the leading edge of astronomy. We are already seeing the first generation of this exciting technology move into civilian observatories with more, such as the guide-star technologies, not far behind.

The Air Force is interested in improved communications that will enhance the capabilities of our space-based assets. One very interesting development program in this area is the application of lasers for very high speed, satellite-to-satellite communications. The inherent advantages of laser systems, compared with conventional RF communication systems, are their small size and most power requirements. This capability will benefit Earth-observation satellites and may be useful on interplanetary spacecraft and future Moon and Mars programs.

One program that will have a major impact on future space projects is the recently initiated Air Force and SDIO program to explore space nuclear power generation. This program is geared at gaining experience with the design and operation of thermionic nuclear power systems. Recently, we purchased two Russian TOPAZ-II nuclear reactors to help us move more quickly into the application phase of this technology. Besides the benefits to military space systems, this technology may enable the powering of direct broadcast satellites and powering robotic spacecraft and crewed bases on the Moon and Mars.

These are just a sampling of the Air Force programs useful to the larger space community. All four Air Force super labs have ongoing technology programs that can benefit a wide range of space programs far beyond the confines of military applications. One continuing challenge to the STIG is to enhance this cross-fertilization of information so that information flows freely among the space community. This SOAR Conference is one excellent means for accomplishing this. However, we need to accelerate methods of data collection and sharing, perhaps through joint technology data bases, so that needed information will be quickly and easily identified and located when it is needed.

In this year when we celebrate the historic voyage of Christopher Columbus, we are really celebrating the human spirit that urges us to seek out that which we do not know—the urge to explore and discover. Fortunately, it is not a lack of spirit that restrains our expanding exploration of the space frontier. The real challenge for achieving our dreams in space rests solidly on the Earth with us here tonight. Columbus, for all his courage and determination, could not have reached the New World without the technologies that made it possible. Neil Armstrong and Buzz Aldrin could not have taken those historic first steps on the Moon without the space technologies that many here tonight

made possible. The challenge of opening the space frontier, to translate the dreams into reality, rests largely on our shoulders.

In this regard, I view this as a most fortunate time to be a part of this nation's space program. The rapidly eroding threat of war coupled with a vigorous space technology base offers a timely opportunity for the governmental cooperation that we have created to move our nation forward in space. We are at a point in time when the technology required to truly open the space frontier is rapidly becoming a reality, thanks to your efforts.

I applaud you for making the STIG a success and thank you for the opportunity to address you this evening.

Technology Transition Panel Discussion

K. KRISHEN: The panel discussion we have for you today is on "Technology Implementation and Transition." This is something that we, in the STIG operations committee, have been thinking [about] for the last couple of years, and it was fortunate for us to have the opportunity this year to include this panel discussion in the SOAR ['92] Program. I would like to quote an excerpt from a recent speech by our NASA Administrator, Dan Goldin. It says [quote], "Missions to the planets will continue. These spacecraft will require the most advanced technologies we can develop. They will require systems that are lightweight, small, durable, and highly integrated. They will employ neural networks and other expert system technologies that will monitor and diagnose and correct problems at a distance. We want to see these technological capabilities adding value to products and manufacturing processes here on Earth" (end of quote). To me, "here on Earth" means we need efficient and safe mission operations for our space programs; [and] "here on Earth", for our private sector, means the technology and techniques [that] enhance our competitiveness in the international market. I would like to share with you a few vugraphs on the dimensions—or the dimensionality—of technology today. (My first vugraph.) When we speak of the space environment, it's a very highly dimensional environment. And, in the past year or so, I tried to research in my own mind what are some of the unique attributes of this environment. As you can see, I ran almost out of the page listing the top-level (we are talking about) environmental attributes such as places in the universe where there is nothing (I don't know how to define that word "nothing" here); places in the universe [that] have unique electromagnetic environments, debris, unique temperature, acoustical environments, magnetism, ellipticity; and so on and so forth. The key, as you heard earlier from Mr. Goldin's speech, is for us to concentrate on lightweight. Weight would be a very high cost for lifting things into space. The key for us to concentrate is to concentrate on size—the volume of things—because, again, it's very expensive for us to take very voluminous things in space. The key for us is also the energy consumption. The less the energy consumption, the better off our systems are. Finally, not the least [that we concentrate on is] the reliability in a space environment. Let me share

K. KRISHEN: (Cont'd) the second vugraph. It has to do with the fact that I was trying to see: What are the dimensions now of the way we express technologies? And, once again I really didn't even try to . . . didn't even get a better feeling of how to express the various aspects of the technology and when we talk about technology. For example, revolutionary, evolutionary, innovative, discipline-unique, time driven sometimes, sometimes resource driven, environment unique. So we have developed a language in itself of expressing the research and technology attributes of certain things. And, these, of necessity—when we are talking about technology—we need to keep this in the back of our minds. The next chart is very generic, and many of the textbooks have it. It depicts the fact that, when you have an evolutionary technology, it really goes in a manner where you're trying to evolve more performance from where you are now. And, when you have a revolutionary technology, you are trying to go in a fashion where you are trying to gain on a quantum jump between where you are now and where you could be if you did evolve that revolutionary technology. To me in the operations world, we are going to have both types of these technologies: both technologies that go evolutionary and the technologies that go revolutionary. And, let me give you an example of these revolutionary technologies. They have to do with virtual reality, as an example, neural networks, fuzzy logic systems, wavelength process and technology—this is a very forceful technology for processing signals, particularly having to do with video signals—LANO technology, which we are seeing in [its] infancy, and superconductivity. All of these technologies will have a profound effect on what we do in the operations world. The next slide I have gives you a feeling of the fact that we can have technology at different stages of growth. For example, we can have the basic concepts, what we call the "Level I" —at NASA Headquarters, we have developed a certain, you can say, "language" in terms of levels—Level I would be the basic research, where you are trying to develop a certain expression or a certain concept based on, for example, Maxwell's equations. And, Level VII would be where you're taking this antenna and trying to qualify it in space. So when we talk about technologies at those levels, it's the fact that we're including both software and hardware.

K. KRISHEN:

(Cont'd)

I wanted to share with you the fact that the number of patents profoundly affects our competitiveness in the world marketplace. For example, on May 28, 1991, as reported in the *New York Times*, you can see in this vugraph the correlation between the number of influential, you can say, patents and the strength—economic strength—and the technological strength of nations. You can see in our case, for example in the US, we're now almost being constant, with a little bit of perturbation up and down. And in Japan, of course, they have been steadily increasing in this influential patents in the technological world. The next chart shows you the impact it has [on] our industry. For example, even in particular sectors of our industry—Kodak versus Fuji—you can see the number of influential patents not keeping up with those of Japan and, naturally, a lack of us being that efficient in the marketplace. So this is the thing we have to keep in [the] background when we are talking about technology transition and technology implementation. The fact that it's not only the space program, but it's the whole entire nation—its economic strength—that will evolve as we try to take these technologies and transition them into the marketplace and into our programs. I wanted to share with you one fact: That is that, if you looked to the scenario of the overall flow of things—for example, the customer could be a program office, it could be industry, it could be our real customer—but we have to have some sort of requirements that come back into the infrastructure or the (what I call) "enterprise infrastructure." That could be NASA, it could be [the] Air Force, it could be McDonnell-Douglas, it could be any of the industries and so on; and then we generate these products and services. The panel today will address this issue of technology transition. How do we take the technology that we have developed and the research concepts and the services that we have developed into that marketplace, into a place of a program where it can be utilized?

We have three panelists from NASA and three from the Air Force, and you will hear first about 10 minutes of discussion from each one of the panelists. Then we'll open up for questions and answers. Let me introduce very briefly all the panelists, and then I will ask each one of the panelists to come without [further] introduction and give their presentation. Mark Gersh is the manager of engineering prototype development for the Space Station Freedom Program,

K. KRISHEN: (Conc'd) Level I, at NASA Headquarters. Steve Riemer is the chief of [the] Advanced Technology Development Office at the Armstrong Laboratory, Brooks Air Force Base. Bob Savely [is the] chief scientist and one of the architects of the SOAR conference; he's the chief scientist for the Advanced Software Technology at the Johnson Space Center, and he works in the Information Systems Directorate. Jeffrey Kantor is the assistant director of the Air Force Human-Systems Program Office at Brooks Air Force Base. Paul Backus is the group leader of the Super-wise Telerobotic Applications Group at NASA's Jet Propulsion Laboratory. Last [but] not least, our previous co-chairman of the operations committee, Melvin Rogers, who is the chief of Plans Division at the Phillips Lab, Kirtland Air Force Base in Albuquerque. And with that, I'll now ask each panelist (in order) to come and give their presentation. Thank you.

M. GERSH: Well, Kumar said we have only about 8 minutes to talk to you. So I've narrowed my slides down from about 60 to 42. I've been involved with SOAR for a couple of years now, and I think this is probably my third tech transition panel. So this time around, I was trying to think about, "Gersh, let's not get on your high horse; let's not get on your soapbox and preach some gospel here. People have heard you enough. And you know, everybody knows about John Muritore's famous quote about 'tech transition being a body contact sport'; so you don't really need to say that any longer. And, work with users and deal with the integration implementation issues." So, I was trying to figure out what it was I was going to talk to you about today; and I was at another workshop meeting that NASA works—a group called the Strategic Avionics Technology Working Group—about 2 [or] 3 weeks ago, and this very issue of technology insertion the barriers, NASA barriers to technology insertion, the group [of] about 200 NASA, NASA contractors, [and] a few DOD people were sprinkled in, and predominately looking at advanced avionics issues for space. But, this is an issue of how do we deal with tech transition, technology insertion? And, as it turns out, what I . . . to be honest, I stole from one of the presentations . . . is, after watching this, I said, "This is pretty good." In fact, I'm not here to preach to you, I'm really here to be more like Arsenio Hall, where each one of these items, they have more left for you to go, "Hmm," to think about it, and figure out, "Hey, this is an issue. What do we do to resolve it?" I used to be in the Air Force, so I'll try to give a

M. GERSH: (Cont'd)

perspective of how some of these relate to what I remember from the Air Force as well. But, the first one: We talked about [a] lack of definition in future programs in time to develop technology to a proper level. It really comes down to, as we're developing our programs and we go through this traditional concept of exploration, prototyping stage, product development, specification, and then pops the product out, supposedly we've got some sort of process in place where we can fully thresh out all the concepts during that concept exploration phase, and that we can do a lot of prototyping. But, it's not working. There're still a lot of technologies that a lot of people will argue are out there that a program like Space Station, for example, should be using within its baseline; but there are barriers to that. So, how can we . . . what can we do to start bringing together some of these technologies in place? An interesting idea is trying to do some solid concept exploration with a lot of prototyping should . . . organizations like CODAR, OAST, and the Air Force Labs—the 6.2 and the 6.3A funding sources—maybe they should become responsible for the concept exploration stage of a major program. Let them be the ones to actually do the prototyping. A recent OAST manager [advocated] the biggest difference between what NASA was able to do in the early 1960s and what we're able to do now is that, in the early 1960s, the stuff was in a sense "one-of-a-kind." There was no split between the research community and the programmatic side of the Agency. Second: Cost and time required to demonstrate. This is really an issue of proving the technology. When I was in the Air Force, I had a General pull me aside—General Ferguson —one day, and he said "Lieutenant . . ." we were on this discussion about tech transfer (I had a couple of projects, and we were trying to figure out how we could get the user organizations to buy into some of this stuff), and he said, "Listen. You've got to realize that you have to prove the technology beyond a reasonable doubt: that a development engineer not only has got to know that the technology is viable, but [he wants] to know what it's going to cost [him] in terms of resources: dollars, people, overall risk." So, what can we do to demonstrate the technology? Within NASA, when it comes to actually demonstrating in on-orbit types of things, the only vehicle we have is Shuttle to really do that, and it's . . . those of you that have tried to implement and fly things on Shuttle and development test objectives and secondary payloads know that it's a very expensive, time-consuming process. So what can we do to sort of release ourselves

M. GERSH: (Cont'd)

from that burden, so to speak, of demonstrating technology? What can we do more on the ground? And, the challenge to NASA is: What can we do to take advantage of some of the things our DOD brethren have been doing in a lot of areas that are very similar? In the areas of aircraft or OPS and things like that. And, not use the simple excuse, "Well, it's a little bit different." Because, when we get right down to it, there're a lot of similarities. A lot of times we talk about having to run in parallel, that we bring a new technology on board and that we'll run in parallel to the baseline operations. And, so we'll do that long enough to gain confidence in the system. We have to do that. You just can't take out the old system and pop in the new. There's got to be a period of time of transition there where you can see the new system run in parallel. But, there is a heavy expense paid for that approach. So what can we do as a collective body to start figuring out ways to streamline that parallel process? Are there things that we can do? Another interesting issue [that] came up is, the world of software, this amorphous blob we call "software" that you can't really touch or feel but know it exists, that you really can't get your hands around it and put your finger on it in the same way that people like control engineers—guidance, navigation, and control engineers—know that they've got a certain set of physical laws they have to operate under, and they know those laws, and they design to those laws, and they can't get away from those laws. So there are no natural constraints affecting software, you know, insofar as development and the test and the verification of that process [are concerned]. So that in itself says, When is enough enough? (Next slide.) It used to be the Government was a driver of various aspects of industry. Well, that's no longer the case. I think that we used to say that the Government drove the software and computing industry in this country back in the 1960s, early 1970s. Well, now I think entertainment's the number one driver—movies and things like that. So we no longer can drive the industry, so we've settled on the fact that we've got to go COTS—commercial off-the-shelf. What can we do, though, to start targeting the ability to go to COTS, to really go to COTS, and say, "Okay, Company X makes this and Company Y makes this and Company Z makes this" and the freedom to be able to go to all three companies? And, this implies flexibility in procurement rules; this implies the overall technical flexibility to sort of swap between a COTS different implementations. Another interesting thing that we probably could borrow from the Air Force is sort of the

M. GERSH: (Cont'd)

design philosophy in a project called SLICE out at Rome Labs, where the idea of trying to pull together a software engineering environment that borrows off, in a sense, COTS ideas, but you center on the interfaces. And, so that you can pop in things that are common to a set of interfaces, and you're not burdened or held hostage to a particular approach. The other issue of standards talks about the evolution of your hardware and software. "Upgradeability" is what I really call it. And, I tell you in Space Station, we're designing an orbiting platform to . . . we used to say "30 years," now we just say "multi-year" . . . But, the idea is, we do have [an] orbiting platform that will be there a long time, and we're designing it with a core data management system that will say, What's sophisticated 3 years ago, relative to what you might have in your laboratory—in a constrained laboratory environment—but 30 years from now, it's not going to be very sophisticated. So, what can we do to start tackling these issues of the upgradeability of systems? And, how do you approach standards from that aspect, so that you can pop in the latest version of high-resolution displays or the latest version of a new processor, or so forth? Another issue is dealing with the cost of operations. Government . . . we always get caught in this game: If we don't spend our money by the end of the year, then that money will get cut and we won't get as much next year. So the same idea sort of holds true here. What incentives do we have to really cut OPS costs, other than the fact that it's just the right thing to do? You know, we in our hearts know that, hey, we're spending a lot of money in Shuttle operations, and we're trying to figure out how to streamline those operations to get the cost down. But, what kinds of incentives could be placed on us? Or what kinds of things can we think about to actually drive us to reducing cost? You know, this really drives home the issue of trying to insert automation technologies. For the longest time, the people in that camp have argued that automation will reduce your costs. I can tell you times that I've sat down with the director of Space Station and said, "If you use this automation in your control center environment, it'll save you money. You'll be able to reduce your people count." And, some of this stuff [has] been going on for a couple of years now, and the director turns around and says, "Shoot, Gene Kranz at MOD hasn't sent us any dollars back." So we know that the reality of the situation says you just don't take out people. Automation probably buys you the opportunity to do that, but what are the incentives for us to really do that? Other than just coming forward

M. GERSH: (Conc'd)

. . . NASA's going through a series of exercises. The Administrator laid down a challenge that said, "Figure out how the various organizations within NASA could live with a reduced budget in the next couple of years." And, because of that threat of reduced budgets, we went and figured out how to restructure, reorganize, reorient some things, and ultimately tried to save that money. In a lot of cases, we're doing things that begs the question, "Well, why didn't we do that before?" And so, Why is itself, just the threat of cutting money, a forcing function? In other words, an incentive dissent is, we need incentives. And, finally, this issue of software contractors: there's a lot of technology out there that always gets argued about improving productivity in software development, saving money in the life cycle (the overall), improving the reliability of your code; and those are all well and good and they're valid arguments. But, really that comes home to, sort of, the next project; in other words, we're trying to incentivize the industry and within Government for the next project. We'll do this stuff; and [on] the next project, we'll save money or we'll be able to do things more reliably. Well, what are the incentives for the current contract? What can we do for a current contractor that we have on task to say, "Geez, there is a way that we can start implementing some of these ideas that we've been touting for a while that save money, save time, save resources." And, again, a lot of these issues are procurement issues. But, the idea here is for you all now to sort of go, "Hmm," think about some things, and really—instead of maybe asking us some questions—you can stand up and fire "what ifs." And, maybe this session evolves into not just six people here on the stage giving you their perspective, but that we really tap into the 250 [or] 300 people that are here.

S. RIEMER:

[In] the next few minutes, I'm going to discuss with you the actual implementation and transition strategies of the Armstrong Laboratory. They're Armstrong Laboratory transition strategies, but, as you can guess, they're also the Air Force Materiel Command's view on transition. I'll be concentrating primarily on the transition of advanced technology development, since that's what I work with on a day-to-day basis. Last week, after I had prepared this briefing, I was looking through the new defense S&T strategy, which was put out on 16 July by DDR&E. And, I noticed that a lot of what I'm going to talk about today was mentioned in that strategy. Specifically it says, in part, that "early and continued user

S. RIEMER: (Cont'd)

involvement is a must, and capabilities should be proven through advanced technology demonstrations." I'll say more about those points in a few minutes. Mark Gersh mentioned, we should demonstrate our technology. You'll see in this talk that a lot of what we do in the Armstrong Laboratory and Air Force Materiel Command in the 6.3A area are technology demonstrations. The title of that slide, you heard this morning by Dr. Welch and also by Dr. Alexander. Since they're both my bosses, I have to mention it as well. There are no—and Dr. Alexander mentioned that to me during lunch today, that make sure you say there are no—unmanned systems. So there are no unmanned systems. The human is the most critical component of any weapons system. And, at Armstrong Laboratory, our technology, they ensure that the people in the operational Air Force are properly selected, trained, equipped, and protected. But, what about all the technologies that really aren't embraced at first by users? We've had some problems with that at Armstrong Laboratory—as I'm sure all laboratories have. And, in the past . . . there are some examples. Like I said, it's nothing new. Parachutes. In World War I, parachutes weren't accepted because it was thought that they would encourage pilots to jump out of the aircraft. Prop versus jet. No one wanted to fly the first jets because [they were] something strange, and they couldn't see the propulsion system. Fly-by-wire: At first, no one trusted computer-driven aircraft. Stealth. Was Stealth fast enough, and could it carry all the ordinance necessary? Space systems. We're still learning about space systems, and that's the purpose of this conference. So what have we learned over the years about getting technology to the user and getting him to accept it? At Armstrong Laboratory, we've had those problems, and I'm sure all of you who work in the other laboratories have had the same problems. The answer, which you're going to hear from me throughout this briefing, is that to ensure successful technology transition, you have to make the user part of the team up front and throughout the S&T and acquisition process. Obviously, if that AFMC scientist at the far right—right there—he obviously didn't get that user right there a part of the R&D team, there are problems. I think it's nothing new. One of General Yates—who's the commander of Air Force Materiel Command—one of his command goals is technological superiority. It's the cornerstone of the S&T process. The policy emphasizes that "technology transition is as important as development of the right technology." Transition planning is as important in our day-to-

S. RIEMER: (Cont'd)

day work as the science. Actually, this technology superiority process is our S&T mission at AFMC and at Armstrong Laboratory, Phillips Laboratory, at Rome Laboratory, and at Wright Laboratory. It says again: Do the right technology, which is a quality and relevant S&T program, and then apply that technology rapidly to operational needs. This chart shows you where S&T (science and technology) fits in the overall RDT&E process, and specifically where 6.3A (advanced technology development) fits. The major goal of 6.3A is to develop technology that is demonstrated—as Mr. Gersh was mentioning—in the user's operational environment and to reduce the risk of technology prior to transitioning to EMD-64 or directly to the major command user. The most important identifier of 6.3A is the ATTD (the advanced technology transition demonstration). In the Army and in the Navy, it's called advanced technology demonstrations; in the Air Force, it's called advanced technology transition demonstrations. This is what I mentioned earlier as part of the core of the defense S&T strategy. Through the ATTD, the major commands see the operational advantages of the technology. And, the SPOs and the ALCs become satisfied with any risks prior to transition. The main point is: We know that the ATTD works in the laboratory, but will it work in the field? Here's some more on the ATTDs. It's a summary slide of what an ATTD is. They got started because of a 1988 Defense Science Board study, which found that defense technology wasn't being transitioned rapidly and should be accelerated. The place to do this, they say, is an early advanced development 6.3A. The technology should be demonstrated at the user's home to show technical feasibility and operational utility. Each ATTD effort should have a documented technology transition plan; and transition planning should involve all parties: the lab scientists, the operational user, the development planners, the XRs in the major commands, and the SPOs. Because of a DSB study, we now have strong personal involvement in management by the technology executive officer—the TEO—at that time Major General Rankine, who just retired. The new AFMC-ST is Brigadier General Paul, and also the Armstrong Laboratory director, Dr. Welch, as well as the directors and the commanders of the other three super laboratories. The second bullet [on the slide] mentions critical experiments. Those are 6.3A programs that aren't yet ready for ATTD status. They are demonstrated in the laboratory environment, with the goal to move them into ATTDs where they [will] be

S. RIEMER: (Cont'd)

demonstrated in the operational environment. As far as resource allocation decisions, the TEO looks at the operational utility assessment scores that the major commands give us and also at how we're executing our technology transition plans. As a continuing tool to assess the results of the technology transition process, Air Force Materiel Command has developed a set of metrics; specifically, how are we in the labs transitioning our ATTD programs? And, by the way, as Mr. Rogers could attest also, we have to report data from these metrics quarterly up to Command, and they report those quarterly at what're called Vision Conferences where all the laboratory directors are present. Therefore, the pressure is on us to make sure that we do the right thing here and get the right technology and also transition it in expeditiously. That overall "big-picture" metric is the expeditious technology transition, and that's tied to the command goal of technological superiority. Supporting that big-picture metric are the TEO metrics: timeliness, which is the ATDD schedule, teamwork, which is the number of technology transition plans we have with the users, the developers, and such, and also those ratings that the major commands give us as far as operational utility. Last year, the AFSC and the AFLC commanders—General Yates and General McDonnell—commissioned a Technology Process Action Team to look at the total technology process, including technology transition, for the new Air Force Materiel Command. That Process Action Team, headed up by Dr. Vince Russo of the Wright Laboratory, found that these here are the best processes for timely and real technology transition (lessons learned, so to speak). All the labs are working toward these best practices. I think, as Dr. Krishen and also Mr. Gersh mentioned, there are problems in technology transition. That's why we have this panel. But, we are now officially striving to get rid of those problems; and these lessons learned, best practices, are in effect now at all the laboratories in Air Force Materiel Command. I think I mentioned these one way or another, but I'd like to just briefly say again what they are. Top management involvement is absolutely necessary. Have frequent reviews of technology transition problems by the team. A continuing process ensures that everyone knows when the technology will be ready for transition. Do the right technology and transition it quickly. Have technology transition plans, because they're the blueprints for executing the transition. And, the user has to be part of the team from beginning to end. Also, measure the results of the technology transition for

S. RIEMER (Conc'd): continued improvement. This last chart summarizes what I've been saying, and it's what the Command and the Armstrong Laboratory and all the other labs are striving toward. First bullet is: We have to nurture the right technologies and apply them rapidly to operational forces. That was a technological superiority goal. This will work, though, only through a strong coupling of all involved throughout all phases of the acquisition process: the users, the laboratories, the development planners, those XR folks, the program officers who offer us their assistance, and the product development engineers. It also includes academia and industry as part of the team as well. And, this is not a system-specific way of doing things. It'll work for space systems; it works for tactical systems; it worked for personnel systems. It's what is absolutely necessary for effective and expeditious technology transition. Thank you.

R. SAVELY: I wanted to approach technology implementation from, perhaps, a slightly different thought process. One is, there are (I guess) three basic motivators in this world: fear, bribery, and reason. [Essentially, bribery is] the most effective means. The first one: Pilot projects, those have been discussed. And, Mark Gersh here, for example, operates a number of those. And, we've recently started another one from the Code Q world, even, where we're essentially paying IBM to utilize formal methods in a pilot study, hoping that they'll build ownership in the use of that technology. So, I think most people are familiar with that idea. IRADs [are] something that usually, I think, probably the room here is split in two. I presume half of us are IRAD evaluators, and the other half are working on IRADs. And historically, IRADs have not been well received when you get stacks of those to evaluate. In fact, some people refuse to evaluate them, I'm told; and I've been known probably to lose a few myself in my youth. However, all of a sudden—bang!—I realized that the IRAD program is a great technology transition mechanism, since it provides the funding to develop ownership of new technology, which is critical for cost-effective implementation. And, those of us who have had experience with change requests know the cost is significant if the contractor is not experienced with the technology involved. In other words, if they're not familiar with it, if they don't like it, you probably can't afford it. Valuable side benefits include, I think, a boost in employees' morale on the contractor's side, which results from involvement in R&D—since many of the

R. SAVELY: (Cont'd)

contract positions involve routine work. And, the sad part about it is that many Government contracts do not provide for an IRAD program, which I believe results in productivity lost from both the failure to use new technology and a higher turnover in personnel. I might point out that a lot of the best IRADs that I've reviewed used NASA technology as a starting point, so I'd like for the contractors—support contractors out here as well as the primes—to keep that in mind. The bottom line the previous speaker addressed the technology transfer activity; primarily, I want to point out that it's an area that cries out for the use of TQM (and we've heard the story that customers need to be involved). But, on the other hand, I urge technology transfer groups to essentially adopt the philosophy of "the customer's always right." Whatever they want, please do it. The next couple of slides come from a recent report that was dedicated completely to technology transfer. In this report, they surveyed a number of corporations that had successful as well as unsuccessful projects. Their summary was, of course, reasonable problems, problems that demonstrate the capabilities of the technology, and finding someone's pet peeve and fix it. I have a couple of examples here I'd like to bring up in that area. One example that was presented at a couple of recent conferences, I think, is pretty impressive. An expert system was developed by American Airlines that only contains 49 rules, and it's called "Hub Slashing"; and it saved $2M the first month that it was used. Hub Slashing is a knowledge-base system [that] recommends contingency plans for American Airline's systems operations control during bad weather or other airport problems where severe schedule reductions must be made. I think these are folks that built a system that saves big bucks and solved a significant problem with American Airlines. So, the next time your flight has been canceled, probably an expert system did it. And, we're probably responsible for that. However, I hope I didn't mislead you into thinking that it'll be easy to transfer your technology into routine use. In fact, one of the complaints that our administrator received on his recent tour of NASA centers from researchers was that their research is not being used. And, he charged them with the task of seeking out test-beds for their technology. I mean, just don't sit around and complain. You have to be in sales and marketing. And, I'd certainly like to reiterate this challenge to each and every one of you, to seek out opportunities to encourage Government and industry to make use of your achievements. And, I think the most difficult task for all of us is to

R. SAVELY: (Conc'd) determine when research has progressed to the point that it can be utilized for operational use, and I believe conferences such as this play a key role in this determination. I suggest that each of you [sets] a goal of coming out of this meeting with at least five new ideas or applications for the technology that you see here in the next couple of days. In summary, I think everyone here should look across to Government for potential users and urge management to fund pilot projects in order for their operational units and their prime contractors to become familiar with the technology. And, in addition, the Government must continually assist IRADs in order to increase the competitive ability of industry, which in turn will make their support of Government contracts more cost-effective and increase their worldwide competitive posture. Thank you.

J. KANTOR: As the only SPO person on this panel, I need to begin by initially clarifying something. SPO does not mean Stupid People from Ohio. Some of us live in San Antonio as well as places farther north where it's cold. SPOs live in a place separate from laboratories, kind of on the other side of the tech transition wall. Our job is to take the science out of the laboratories and turn that science into fully supportable user systems. It's a job that's challenging, different from the laboratories, and I'd like to give you the SPO perspective—or at least my personal SPO perspective—of how that process might be improved. The first place to look is to see how the research, development, and acquisition process is structured in the Air Force right now. There're basically five phases. The first three—the R&D phases—are equated with 6.1, 6.2, 6.3 kinds of work (as Mr. Riemer has talked about). That's done by a laboratory. Products then typically go from the laboratory to a system program office (a SPO) and into the acquisition phase. Those phases are engineering and manufacturing development—it used to be called full-scale development—and then a production and fielding type of activity. In fact, under Air Force Materiel Command, fielding is now considered to include operational day-to-day support of fielded systems under a concept called integrated weapon system management. For instance, within our SPO, we are the single point of contact from the beginning a life support system piece of equipment or system hits the engineering and manufacturing development phase until it is retired some 15 or 20 years later. So, we do both the engineering manufacturing and development fielding and, through an office at Kelly

J. KANTOR: (Cont'd)

Air Force Base, the day-to-day support of life support equipment. And, you'll see more and more of that because it's one of the cornerstones of how Air Force Materiel Command wants to do business. The problem with this type structure is you have some built-in difficulties in addition to the ones that crop up from other sources. The first is just due to organizational boundaries. You have a laboratory doing some of the work, a SPO doing other parts of the work, and they have inherently different goals and subcultures. You have dissimilar focuses in the organizations. People in the laboratories rightly want to push the state of the art. People in the system program offices are basically interested in managing risk. They have to meet cost, schedule, and performance milestones and requirements that are equally equated—you can't breach or break any of them, or else you wind up explaining to a Brigadier General or above why you've done that and why you're not doing your job very well. So, very different orientations and focuses [exist] in the two different organizations. And, different kinds of skills are involved as well. There are basically three types of problems that we see develop in technology transitions from the lab to the SPOs. The first is in terms of the type of documentation that we typically receive from the laboratory. They are scientific reports; they are good technical reports; but they don't give us as much information as we need to start engineering the system for the things like supportability and maintainability and systems safety—those kinds of things. It was pointed out to me earlier today, that's probably a Freudian misspelling of "ill-at-ease." (I'll leave that to your own determination.) The other thing that documentation typically lacks is a requirement given to the acquisition community by the Federal acquisition laws that we have to provide enough detail for contracting so that any competent contractor can submit a good proposal and price out the project. That is substantially more documentation than labs ever produce, and it takes a while to go from what the labs give us to that level of detail that we need for contracting. Another problem is that labs have been working on this research [and have] built over time an incredible store of expertise in that particular area. That expertise typically does not transfer to the System Program Office. We have to start with a much lower level of technical expertise in that particular area. And, finally, there is a lot of difficulty often in building a focused transition team. We need people from the laboratory to help us early on. It's difficult to get them to be interested in the same things

J. KANTOR: (Cont'd) that we in the program office are interested in. Scientists are interested in doing science; program managers from the SPO are interested in putting together a fully supportable system. It's difficult to get those two different sets of interests to work together well. How can we improve that process? If we take a look at what the three main components of technology transition are, I think we may get an idea. The first is the technologies themselves; and that's typically not a problem. Our labs produce truly state-of-the-art technologies. The next component is a process for doing it. And, [although] this is probably heresy in a bureaucratic organization, I think that's probably the least important. The key to success is the people. Technology transition to me is kind of a strange term. It kind of gives me the impression that somehow you take technology, you put a key in it, wind it up, and it happily runs down the various phases and out to the users. That's not what happens at all. It's people who, throughout the process, transition it from the lab, through the SPO, to the user. And, I think people are the key to improving technology transition. In fact, once you realize that and begin to deal with technology transition as a people process, there are some fairly straightforward kinds of things you can do. How do you improve people's performance in technology transition? The same way you improve them in anything else. Train them to do the job, motivate them to do the job, and then make it as easy as possible—facilitate their performance. I've been involved in Government research in one way or another for 17 years; I've never had any kind of formal training in technology transition. There are courses that we have in science and technology, courses that we have in acquisition, [but] no courses to teach you how to do technology transition. To me, that's an obvious gap in the kind of training we're providing our people. All members of the team have to understand what's involved. The second item is one of some sensitivity to some people; but, to me, you have to move people around to different jobs. It was pointed out to me earlier today that I'm probably kind of an odd individual—odder in some respects than I'd like to talk about right now. But, odd with respect to the fact that I spent my first 12 years in a Government laboratory—in the Air Force Human Resources Laboratory—doing scientific kinds of work. I had a great time; I enjoyed it. From there, I moved on to a development and planning office—a Headquarters Product Division XR kind of job—[and] worked there for 4 years. And, in the last year and a half I've been assistant director of the System Program Office. So I've had

J. KANTOR: (Cont'd)

the opportunity, through moving around from one job to another, to see what people do in the lab, what people do in development planning, and what people do in the System Program Office. I should not be that unusual—with some respects. But, I should not be that unusual with respect to the kind of job experiences that we're growing in individuals that we want to see help us transition technology. And, finally, we need to train people to work together. There's a lot of room for doing total quality type of team training for technology transition. One of the things I think we need to do is identify whose job it is to technology transition. The people in the labs, that's not their job; their job primarily is to do science and technology. People in the SPO; their job is program management or functional engineering. We can't usually point to one individual and say, "You are the one who transitions technology." We need to make that somebody's primary job responsibility in the system and then directly reward that person for doing a good job. And, finally, develop and track metrics. A common expression going around the Air Force today is, "If you're not keeping score, you're just practicing." And finally, we need to facilitate tech transition. We need to establish feedback loops. At one time, bring in people from the using community, the laboratory, and the SPOs, sit them all down at the same time, and have them work through some of the issues. We, within Air Force Materiel Command—within this integrated weapon system management program—have a bulletin board that we can all electronically tap into with lessons learned. So, I can sit down at my computer screen in San Antonio and, without any problem at all, tap into the collective mistakes made across the entire Command. This is a wonderful opportunity for me to be creative and make my own mistakes, rather than [to] make mistakes other people have already done. We need to also ensure that the kinds of tech transition plans we develop are executable. To paraphrase an old expression, "We don't want to try to cram 10 lbs of transition into a 5 lb plan." And, in particular, we need to make sure the schedules that we set up for tech transition are realistic. The program director in my SPO says, "It's a lot easier to have to stand up in front of your user and say, 'I'm sorry I'm delivering early' than [to say] 'I'm sorry I'm delivering late.'" And, finally, we need to establish a graveyard pool. Tech transition is not an everyday occurrence. It happens a couple of times in any one person's career. So we need to capture the group of people who have done this and have them, perhaps, do some kind of advisory

J. KANTOR: (Conc'd) board process to young people who are doing it for the first time. Kind of a quick summary: There's a wall between research, development, and acquisition. It's higher in some places; it's lower in [other places]. We need to have people work to bring that wall down. The way to do it is simply to give them a reason to bring the wall down: motivate and reward them, give them the tools, train them how to do it, and then make it as easy as possible—facilitate the transition of technology from one organization to another and, eventually, to the user.

P. BACKUS: . . . is the involvement of people and understanding their motivations. I've been involved in a tech transfer on two occasions. One in transferring robotics technology from JPL to Goddard Space Flight Center. And, now I'm involved again in tech transfer in transferring robotics technology from JPL to Johnson Space Center. So, I'm actually in the process right now, and we're grappling with all of these problems. But, actually we have a job to do right now, and we need to solve these problems and understand them ourselves so we can do the job right now. The goal in the technology transfer, if you wanted to put it simply, is to transfer the technology from the technology group to the product. There may be a lot of steps in between—hopefully not so many steps in between—but you need to keep your eye on the prize. The prize is the actual transition, the product being better because you did it than it would have been otherwise. You need to keep focused on that goal. There are a couple of approaches to the technology transfer. One might be to transfer component technologies to the product. The other would be to develop a complete prototype and transition that to a development organization. Some critical factors in the technology transfer process: First is to have clear objectives, a schedule, and funding that are realistic. And, then [second]—and which is actually the focus of my few slides—is that the attitude of the people really affects the technology transfer. As you've just heard, he said technology transfer is a "people process"; and that's really what you have to understand to develop the right system to get the technology transfer. I could talk of the mechanisms, but really you need to have the desire of the people involved to get the job done. They need to desire to work as a team; and, to do that, really you need to provide incentives to these people to work as a team—both as the technology developers and the end users. And, they need to be strong incentives to really get the job done correctly; and, if necessary, you may need to

P. BACKUS: (Cont'd)

provide some attitude adjustments. Here is a list of a few of the factors that create a positive attitude. It's important that you . . . to do the tech transfer process, you need to create a positive attitude on the people developing the technology and the users who will take that technology. Again, I'm stating that realistic and clear objectives, schedule, and funding [be] well understood by both groups. In the technology development, you have to involve the users early on—I think everyone here has stated that, and as a technology person I agree with that. And, they should be involved early on. We should involve the users early on, but also it's important to involve the technologist early on in the product requirements and definitions so that the product is designed at the outset to be able to incorporate the advanced technologies to make it better. And, not just try to incorporate a new technology on somewhere far along in the process—because it may be much more difficult at that time to transition the technology into the product. It's important that clear responsibilities [be defined], both by the technologists and by the users, so the technologists know what they're supposed to do and they also know what the users are going to be providing to them (and vice versa). It's important to have open communications between the technologists and the users, so they're interacting on a daily basis . . . not a daily basis, but on a frequent basis. You can do your best to plan ahead, but really—as they say—it's a body contact sport. Things are dynamic; the product is being developed; you're learning as you're going; you need to keep the constant communication between the groups. You need to have . . . well, it's desirable to have the compatibility of the hardware and software. If you're developing that piece of the technology [that] is software, it needs to be able to transition into the product—into the user. So it's useful at least to have the same language but, hopefully, more than that—the same interfaces, the same process in the software development—and those things need to be discussed at the outset. And, also, people need to have the attitude where you do what it takes to succeed. So it's important that you have a goal and then you do what it takes to succeed. The goal has to be realistic, but you have to be able to be flexible along the way so that you really do what it takes to succeed to meet that goal. And, then an important thing is to minimize the intermediate organizations to the product. Your technologists are developing technology. How much do they feel responsible for transitioning that technology to the product? If the technologist knows that there are

P. BACKUS: (Cont'd)

going to be many steps between the technologist and the product, then the technologist might not feel as responsible for being relevant in the technology development. But, the technology person doesn't know that he really doesn't have to put it into a real product. We need a measurement of success for the process so you know that you're going toward that goal. You have a goal, and you have a measurement of success for that goal. And, as I just heard and I'd like to repeat, it's important to be able to directly reward the tech transition. If you have been successful in technology transfer, is there some reward in that? Is there some incentive for doing successful technology transitions? In the next slide, I have some factors affecting the technologist's desire to transition the technology. One is, would the technologist rather play in the research sandbox, which has beem a complaint in the past (and sometimes probably true)? It's important to provide strong incentives to the technologist and research committee to do relevant technology. And, the factor might be from the technologist's point of view: Are they transferring technology to a competing organization? They might be concerned about, What happens after the technology is transferred? Is there a follow-on project to be able to continue to develop that technology? Or does the system stop at that point? Is there some benefit to have been successful with developing the technology and transitioning it? Is there some benefit for continuing work after that? Should the technologist really feel accountable to transitioning that technology? Should [the technologist] feel accountable for the technology being relevant when they make it? Should he feel accountable for the product working at the end? Incentives need to be there so that the technologist feels accountable for the process. And, the technologist might be concerned about whether technology is really going to the product or is it going to be reworked? So that goes back to: Do you feel responsible or not? If you know that it's going to be reworked several times along the way, then you know that your work is not going to go directly to the product. So then, there are some problems with your having some incentives for making the products better. And, on the other side of the fence, there are factors [that] affect the user's desire to receive the technology. One common one is the "not invented here" syndrome. Again, we need incentives so that the user organizations want to incorporate the best technologies. So those very strong incentives need to be put in place so that they want to incorporate the best technologies. They may be concerned about, Are they competing

P. BACKUS: (Conc'd) for the same R&D dollars that they're getting the technology from? And, there's the issue of accountability of the intermediate development or integration group. If they're taking technology from a research organization and then reworking it to go into a product, what is their responsibility? Are they really responsible for making the technology work? Was it the original research organization? So, at that point, there could be some confusion as to who is really responsible for getting the final products to work. And also, there might be the question with multiple, intermediate groups: Who is the user? When you're transitioning technology to intermediate organizations, is each intermediate organization the user? Or is the product the user? And, from the technologist's point of view, it's not always clear who is the user and who [you are] really targeting. And, then there's the issue of [does] the end user really understand the technology, the capabilities of the technology, and the use of that technology? You need the teamwork between the technologist and the user so that the technologist understands the users' needs, and the users can understand the capabilities of the technology and then work together to get those technologies to work in the products. This issue [also concerns] the users: Are they ready to accept the technology, or is it going to be put on the shelf to be used at some later date? When it's put on the shelf and the technology is transferred, there's the loss of knowledge base in the people that have developed the specific technology and the loss in the training that's occurring during the teamwork of the technology transition. So it's important to be able to develop the technology and first transition it into the product so there's not a large time lag where you lose that knowledge base in the people that are involved. In conclusion, and as I stated [before], the attitude of people that are doing the transition—both the technologists and the users—is a key point in the technology transfer. And to create that attitude, sometimes you need very strong incentives to get it to work.

M. ROGERS: The position I feel like I am in is: I could say "Ditto" and sit down right now. I think just about everything you've heard just about every perspective on technology transition you can. You're going to get to hear another one. The title I put on this basically could have been the last slide in the presentation on tech transitions from our perspective. I am lucky to be last. There have been two

M. ROGERS: (Cont'd)

other Air Force folks precede me [who] have basically laid all the groundwork to a lot of what we do. As a good many of you out there know, we work to the same beat—sing to the same tune, basically. The reason I put this slide in is for . . . well, there are a variety of reasons. I wanted to preclude any mistakes that there might be an identity crisis vis-à-vis the petroleum industry [at] the Eber-Phillips Laboratories in Bartlesville, Oklahoma. And, this Phillips Laboratory is principally located at Kirtland Air Force Base with directorates—technical directorates—at both Hampstan and at Edwards as well as some remote sites at Mallory and in Florida. The little circles up there were intended to convey a variety of messages, not the least of which is a strengthening personnel base and a strengthening budget, almost as we speak. However, for this fiscal year, the Phillips Laboratory does work to approximately an $800M budget, 45% of which is Air Force S&T, another 35% or so which is SDI, and the "other" is other. And, approximately 2220 people—until they hear what I did here, and then it might be one less. Our tech base works in [the following] arenas: geophysics, space and missiles, [and] advanced weapons. I don't intend to go into this. I just wanted to establish the framework that the Phillips Laboratory used the rest of the world and technology transition from. This is kind of what it comes down to. This is a bad example of tech transition. They invented it, but they didn't run out carrying any of it. The point is, with apologies to Larsen, that you do have to do more than just invent it in order to stay in business. I've noticed, basically, the Air Force presentations have been fairly careful in the selection of words between "tech transfer" and "tech transition"; and that, when I got the opportunity—as it were—to participate in this panel, that's one of the first things I thought I'd better do. And, a dictionary doesn't help an Air Force guy in establishing his position. You go to an Air Force RIC; and tech transfer is specifically to get "enhancement or promote innovation for commercial or public purpose." That leaves out a lot of what DOD can end up doing, of course. And, when we transition from one acquisition phase to another, get it into a system, and principally that system is going to be a military system. And, that's tech transition, and that's basically the piece of the story that I wanted to relate to you—and with these ground rules. You've heard from two previous Air Force speakers about the science and technology strategy and new acquisition strategy. The reference to basic research is the 6.1

M. ROGERS: (Cont'd)

funding; the exploratory development is the 6.2; and the technology demonstrations, be they critical experiments or advanced technology transition demonstrations, is the 6.3A funding. Sometimes those go on the shelf. Sometimes they go directly into a system at any point, and it may go into a system at any point in its development. I added another circle. This chart is an OSD chart describing the current environment in terms of S&T strategy. I added the tech planning team. And, that's sort of going to be how I bring our interface with the user for the transition into the discussion. The names keep changing, as they are in the process of being established. General Yates, a year ago last April (I believe it was), endorsed the formation of what at the time was called "technology oversight centers." Now, through a process of... a bureaucratic (shall we say?) at the very least, there has been a variety of shifts in terms of what they're called. As a matter of fact, the current name... the name I will use through here is "technology planning teams." Each team is expected to consist of basically lab representation, the development planner—which would be, in the Air Force parlance, the development planner is the "XR" at the product center—the SPO—and you've heard the SPO side of the story—and the user. Now sometimes the user is the SPO; sometimes he is in operating command; sometimes it's all of the above. It depends on where you might be in the process. What I've listed are the four product centers for the Air Force, and, at least at one time in fairly recent history, which tech planning teams each of them were responsible for. These are sometimes functional, sometimes mission oriented—it depends on the center and [its] primary customer. The center that Phillips Laboratory is principally involved with is the Space and Missile Systems Center, which used to be known as Space Systems Division, out of Los Angeles. They have four teams. Those teams are indicated here. I, again, have no intention of going into the details, but just to show you that there is a process and we are expected to participate in it in the course of transitioning technology. Those team participants, which include the lab representation and the user (in this case, it might well be the Air Force Space Command for the Space and Missile Center), work to develop a 20-year plan and put the plan in process on the basis of working from the strategies required, to the task required to do the mission, and ultimately system concepts—after you've identified deficiencies—to cover the current and the projected deficiencies.

M. ROGERS: (Cont'd)

And, more, the technology—the supporting technology—has to come into play. This is our primary opportunity to insert, to transition technology from one acquisition phase to another—from a 6.3a program that supports a noted, duly road map (if you will) deficiency and get it into the pipeline where it can in fact become implemented as a strategy in support of the strategy in national defense needs. Then down at the laboratory level where I spend my days, we have to take into account that the strategies to task, the other players, and the tech planning teams and implement all that and get the word out on an annual—at the very least—updated on an annual basis. And, that annual basis is a part of producing the three plans that I breezed over right up front: the technology area plans that each laboratory is responsible for. The insertion opportunities that we have to account for in terms of doing our plan are indicated early on. For instance, in the October timeframe, I have subtitled it "an idealized chart," because frequently the planning for the planning gets behind. But, the intent is to bring the users, the tech planning teams, other laboratories, industry, NASA, etc., bring their inputs to the table at least on an annual basis in the October timeframe in order to work the process that comes out with a budget . . . aligned with the budget plan that's got a 20-year vision and a 1- to 2-year implementation, if you will. I've taken one piece of that plan and emphasized it, as it were, in terms of the tech transition planning. And, this is the annual cycle where, in fact, we bring in the update from the outside world, do both our own planning against it [and] do a sanity check, and rotate it through a tech planning team in order to assess where and how it fits [in], and where does it fit with the other outside world priorities. But, ultimately then, the final decision is made by a laboratory commander (with some help, as has been indicated by the other Air Force speakers) with regard to metrics that our commander reports on on a quarterly basis to his boss. Last year, it was General Rankine; this year, it's General Paul; and then, of course, General Paul reports on to General Yates relative to our success in tech transition. Last . . . well, it was earlier this year, actually, Major General Rankine (before he retired, which was in the early June timeframe) wrote a position paper, which he passed on to General Yates, which General Yates has been using in defense of the laboratories at the OSD level. And, General Rankine said, "There are two main roles for the military laboratories." Those are shown at the top.

M. ROGERS: (Conc'd) [The] first is, you have to develop the right technology; and the second is, to facilitate the rapid transition of that technology. Developing the right technology, in General Rankine's terms, reflects a lot of the policies and decisions, and some of the previous verbiage you've heard from the other speakers here are the quality of it and the relevance of it; and we're graded on that. "Facilitating the rapid transition" is picking the right ones and then backing them. And, "the process that's in place" includes a vote by all the users and then a rack-and-stack (if you will) in order to assign the somewhat precious resources that the S&T community has within the Air Force to those [projects] that have the highest priority in the users' eyes. I've got to point out that the word "rapid" is a relative term—it's kind of like Jeff implied earlier; he indicated that we were lucky to have two Phase II transitions in a given career. A couple of examples [are]: The top one on Peacekeeper was 1974, when the transition process started. And, it was completed, essentially if you will, in 1992 when our Commander-in-Chief and President removed the approach with regard to the peacekeeper in the field and the number that were going to be there. The other one that's cited—no detail, we'll go into—but that one started as a tech push item in the 1980s timeframe, and it's still being worked; although some of it was fielded in Desert Storm. This again is, with attributions, not with the graphics but otherwise to maybe General Rankine and his position paper. He said there were three conditions that were necessary for technology transition. The user has to want it, that means conceptual design, analysis of payoffs; he's got to accept it—the weapons system program office, the SPO as it were, has to accept it, it has to have been demonstrated sufficiently that, with risk reduction, he's willing to incorporate it; and, one of the biggest holes in the tent is the contractor—industry has to propose it. If they don't propose it, if we haven't worked with them sufficiently in the process of developing it, it isn't going to get transitioned to that final step. That concludes my presentation.

K. KRISHEN: You heard some views on the technology implementation and conditions from our six panelists. As I was listening, I wrote a few terms I'd like to share with you. These are not necessarily the key points our panelists made, but I thought just to give you an idea of the different things that were said here. For example,

K. KRISHEN: (Cont'd)

Mark Gersh: "body contact sport; incentives." Steve Riemer: "operational environment." That's a key thing in general, you know; if the technology has to be transferred, it has to go in an environment and we need to define that environment. Bob Savely: "bribery." I think that's something for us to think about, what he meant; but that's something that came loud and clear to me. Jeff Kantor: "development of personnel for tech transfer, tech transition, and a rewarding system." Paul Backus: "interaction, attitude, people—people the key." Mel Rogers: I thought I heard him say "planning process," and I think that was another dimension he added to the list that we have here. I had developed a chart, and now I'm a little bit shy to show [it to] you, but I'll go ahead and share it with you. And, this was the chart that previously we just quickly flashed. It has to do with a challenge for us. And, what I thought was that, as operations people—people concerned with making efficient operations—if you look at technology transition and implementation as a (quote/unquote) "operations type of thing," then these might be relevant and meaningful. The first one here, the challenge for all of us, is to develop efficient research and technology implementation processes and infrastructures. I've seen one example, and I'm very pleased with it. Dr. Paul Chiu, for example at the University of Houston, after he did his pioneering work on superconductivity, he has put together really a very good institute to advertise the different aspects, starting from research to technology. So, I think we need to think about this process, how we're going to bring these seed ideas and take them to fruition. And, definitely we must utilize the graybeards, experienced people, and we must definitely take care of lessons learned. That's the key item here when you are developing these plans. The second point that might be relevant is that we need to implement processes. Now, I heard people are more important than processes and I agree. People make processes, in my opinion. But, the processes will make this transition of technology possible. And, the processes cannot be great necessarily to start with, but we have to have again a mechanical processes revision to continuously improve these processes. One of my colleagues keeps saying, "Aggressively, continuously improve." Whatever you like to use. But, keep improving this process of transition. And, in general, we must develop this process; the fact that it's a part of the program itself—it doesn't skip in a puddle of vast somewheres. It's a totally integrated

K. KRISHEN: (Conc'd) part, from the very inception of a project or a program. We have a challenge, all of us, and let's see how we can address that challenge. The challenge is that, out there are many more viewpoints residing in your knowledge base. And, I want to challenge you to come over here. Unfortunately, our video will not tape different directions. And, if you please, share with us some of your ideas. While you are thinking, let me quickly also finish one piece of business. We have a legitimate question that has been posed to one of our panelists, who doesn't know it's coming. It's Steve Riemer. And, the question is, "What is being done —specifically by you—to transmit in layman's terms the technical transfer activity and results to the media, congressmen, and general public?"

S. RIEMER: Like I mentioned in my talk, we all have requirements that Materiel Command has laid down upon us. We have to report up to them. My boss, the director of Armstrong Laboratory, gets information from me, so he'll know what General Yates knows about our laboratory. The metrics I talked about before—the measurement of the technology transition process—that's documented in many, many ways. It isn't just charts of how many technology transition plans you have [or] how many technologies have been inserted. Not what the major command has given you as far as operational utility assessment scores. But, it's a whole other type of documentation, with words and such that are passed up to Materiel Command Headquarters. We personally don't give Congress those words. We give Materiel Command [the words]; Materiel Command gives the Secretary of the Air Force, AQ (Staff QA) those words and the data from those metrics. I think it's their responsibility to ensure that Congress sees it. Now, that stuff is in the descriptive summaries that we have to prepare that goes to Congress and other types of documentation. So it is documented—the technologies that have transitioned and what the operational uses of those technologies are. But, we specifically do not prepare metrics data that go specifically up to Congress. I think the other part of the questions was to the media: How do we document what we've done to the media? We put out all kinds of reports at Brooks Air Force Base, as I'm sure the other laboratories do with their public affairs folks. That information gets in the newspapers, in the news . . . I think General Yates, at the Air Force Association (I think) Conference a few months

S. RIEMER: (Conc'd) ago, had picked for the media certain scientists who display their technology and to talk about that technology and how it has transitioned. You'll hear a speaker (I saw him on the program at one of the panel sessions) named Wes Region, I think he's going to speak tomorrow or the next day on virtual environments. He has transitioned a lot of technology, and I think that you'll learn something from him; and the technology that he has worked on and has transitioned has been written up in San Antonio newspapers as recently as a couple of weeks ago. So, formally we do not pass along that information. We pass along the information that we have to up to our Headquarters, and those folks, I hope, are responsible enough to pass it on through their chain to the congressmen and to the media and such.

K. KRISHEN: Thank you, Steve. Do we have anybody [who] would like to come here, please?

J. MALIN: I know I'm not a graybeard, but I qualify as a "grayhair." I've been working in the technology transition, or technology integration, trenches quite successfully for a number of years now. And, I was very impressed by Mr. Kantor's description of some of the problems, which I do agree with, and I'd like to put them in NASA terms if I might. I think that it's very important—in fact, it's critical—that we define and fund the technology transition role. Technology transition role includes technology integration activities and risk-reduction activities. Current NASA technology readiness levels are misleading, because they define the role instead of [conducting] yearly test validation and demonstrations. This ignores the substance of the role, which is integration and risk reduction. I've been doing it very successfully. I'd like to have it recognized and funded.

K. KRISHEN: Thank you very much. I'm going to ask Mark Gersh a question. Jeff brought up the question of . . . or proposed that we develop personnel for this technology transition. Do you know of any plans within NASA where people are taught in those terms? That we have a set of people that are just addressing that technology transition? I just want to see if there was anything being addressed.

M. GERSH: That are specifically trained or some sort of training program that will target tech transition as a subject?

K. KRISHEN: Yes.

M. GERSH: No.

P. FRIEDLAND: Just a couple of brief comments. But, one thing that Jane said, I think it's true, but I also think that all the people that I know . . . I mean, most of the engineers and scientists I know at NASA, if they're not interested in technology transition, it's a little hard for me to understand why they're at NASA—unless jobs are scarce or something like that. Because it seems to me the major incentive we have within the Agency for keeping really good people around is the opportunity to see their dreams implemented within something as exciting as space. Although I certainly agree whole-heartedly that rewarding people adequately for that is absolutely right. Actually, one of the comments I wanted to make real quickly has to do with that and with Bob Savely's advice about bribery; we could call it "incentivization," if we want to be more polite. But, actually I learned recently that NASA has an incredibly powerful incentivization program for tech transfer—it's called The Space Act Award. And, those of us in the Government know it's a little hard to give somebody a real reward for doing good work. I mean, a couple of hundred bucks is a big performance award. Except it turns out this National Space Act thing has awards set up, and those awards can go up to like $20,000 or $30,000. And, I recently nominated some people in our group who had built a system called "Autoclass," which has gone out to industry. And, you know, you put in this award and they interview from industry who are using it. If enough of them say, "Yes," they get this big award. So I was real happy to see Gayle give our team—at no cost to me—like a $10,000 personal award to three or four people. And, that's real strong incentivization, which apparently not enough people at NASA take advantage of. So, that's just a plug for using mechanisms that we already have. The only other comments I want to make, which is brief, was to comment that, of all the comments you've

P. FRIEDLAND: (Conc'd) heard, you people should remember that, in many situations they'll be true, they'll be very true, and in many situations they won't be. My own experience on lots of technology transition projects was that flexibility is really the key. There's one underlined lesson that a couple of people mentioned that is 100% correct, which is that: It takes place at the interpersonal level. And that, getting the people doing the work and getting the real users—I mean, the people who really will be using the stuff at their consoles or out in the field—together early and making it, empowering them—to use some other good words our current boss likes us to use—to do the things, is the thing that I have found the most powerful. The only other comment I'll make is that, I saw one thing [that] I think in at least 50% of the time isn't correct, which was to involve senior management early. In some situations, that may be correct; but in many situations that I've seen, involving senior management early may slow up the technology transfer process by 3, 4, [or] 5 years, decades, centuries, millennia . . . Not because senior management is dumb, or anything like that; but because the agendas about things one has to worry about when one's responsible for whole NASA centers or codes are such that make the kind of wanting things down on paper and risk reduction necessary, which don't necessarily speed up the technology transfer process. So I'll repeat a quote that Grace Murray Hopper used to give [that] is her goal for her people's success in the Government, [which] was, "Do something that makes me look good and then you can make excuses for why you did it the right way later."

T. CALLAGHAN: Just a quick comment. You said that there was no real discipline addressed at technology transfer. I think human factors is that discipline, or at least a lot of aspects of it. First of all, it's a discipline that people need to understand all the phases of development so that you can act as an interpreter, from the scientists down to the technology people—the people who are actually applying the stuff. People in human factors have experience in measuring difficult things such as attitudes. Also, they have a basic knowledge of . . . understanding of human physiology, scientific method, and engineering. So, I think it's a discipline that, if nothing else, you can borrow some of the tools that we use and maybe even expand on human factors at the same time.

K. KRISHEN: I really appreciate those comments. We still have a little time for another viewpoint, if there is one. If not, I'd like to ask . . . Let me go ahead and ask a question. Again, probably we'll go to our Air Force colleagues because I just explained a little bit about the operational environment for NASA. I'm just wondering and personally am curious, how do you define operational environment for DOD? Maybe, Steve, you had the previous one of the difficult questions; you can answer this one, too. You use the words "operations environment"; what did you really mean?

S. RIEMER: When we say operational environment, [it means] you're developing a technology for Air Training Command. You tested at Air Training Command. I think Peter mentioned at the consoles, the user's environment: where the user works; at his home. What used to be Tactical Air Command is now Air Combat Command. You test it on the range. You test the technology in the user's operational environment. You don't test it in a laboratory. You take it out of the laboratory and have [the user] actually use it more or less as a prototype.

K. KRISHEN: Let's give a hand to the panelists. Let me also indicate that we are videotaping this; and, if you feel that your organization can use this sort of videotape, please let us know.

SECTION I

ROBOTICS AND TELEPRESENCE

Session R1: SUPERVISORY CONTROL

Session Chair: Dr. Charles Weisban

in Proc. of SOAR Conference, August 4-6, 1992, Houston, Texas.

An Iconic Programming Language for Sensor-Based Robots

Matthew Gertz, David B. Stewart, Pradeep K. Khosla

Department of Electrical and Computer Engineering
The Robotics Institute
Carnegie Mellon University, Pittsburgh, PA. 15213

Abstract— **In this paper we describe an iconic programming language called Onika for sensor-based robotic systems. Onika is both modular and reconfigurable and can be used with any system architecture and real-time operating system. Onika is also a multi-level programming environment wherein tasks are built by connecting a series of icons which in turn can be defined in terms of other icons at the lower levels. Expert users are also allowed to use control block form to define servo tasks. The icons in Onika are both shape and color coded like pieces of a jigsaw puzzle thus providing a form of error control in the development of high level applications.**

Keywords- Programming, Real-time, Iconic Language, Sensor-based robots, visual programming.

1 Introduction

Programming manipulators to perform tasks can often be a difficult and frustrating task. Many of the programming languages available have a C-like syntax, which makes developing applications very difficult for persons not having an adequate background in programming. Deciphering and debugging cryptic, non-portable, and ill-commented code can waste many man-hours of valuable time, while executing such questionable code can be physically dangerous to both machine and operator. Man-hours are also wasted when an operator must undergo lengthy training to be able to operate a robotic system. While C-language programming is appropriate for experienced programmers, it is inappropriate for users who are interested only in effectively using a manipulator or a robotic system. What is, therefore, needed is a programming language and a system that simultaneously provides the ability to develop systems level code and also allows users to program applications easily.

At Carnegie Mellon University, we are pursuing research with the goal of creating a programming and control environment, for sensor-based systems, that allows for rapid development of applications through automatic generation and validation of real-time code. In order to make sensor-based robot systems easy to use and program, we are also developing an *iconic programming language* (IPL) called *Onika* for use as a *human-machine interface* for programming robotic systems. In this paper, we provide an overview of the current version of Onika and its capabilities for programming sensor-based systems.

Onika is a multi-level programming environment that is both modular and reconfigurable. At the highest level applications for a sensor-based manipulator[1] are created by choosing icons which represent objects, jobs, and expressions, and arranging them in a logical sequence. At lower levels, robotics-savvy users (or experienced programmers) can additionally define jobs, new icons, etc. for inclusion into the applications. At the lower level, it is also possible to combine icons representing tasks into control-block form and to bind C-language code to an icon. Once a task level iconic program is created in Onika, the underlying system provides the capability to develop the equivalent C-program for execution. The underlying system consists of the reconfigurable software structure[10] and the Chimera real-time operating system that we have developed [3].

In contrast to the previous work done on VPLs, described in the next section, we are developing our IPL to be reconfigurable, customizable, and able to sit in any architecture, drawing upon the resources of the resident real-time operating system (such as OS-9, CHIMERA II, VxWorks, etc.). The icons in Onika are both shape and color coded and can be thought of as pieces of a jig-saw puzzle. This is advantageous because a novice user cannot connect completely arbitrary icons to develop a program, at the task level, that does not make logical sense.

Onika also permits smaller sets of iconic programs to be combined to create larger programs (represented by one icon), thus allowing for the rapid creation of a library of tried-and-true procedures for the manipulator. Furthermore, by loading in the specifications of any particular manipulator, it is planned that our IPL will be able to run its programs on different manipulator systems *without the user needing to rewrite the code.* Translation would be done at a lower level in the system architecture, with the help of a *specs file* created for each manipulator (which would list construction data, joint limits and lengths, moments of inertia, and so forth). Finally, development of a lower-level simulator for the IPL is underway, so that an application created for a manipulator can be simulated in real-time without changing the application at all -- from a telerobotic standpoint, there would be no difference in the type of information received and the amount of control exerted in a simulation as opposed to an actual run.

[1] Throughout this paper, we shall use the term *manipulator* to refer to the system programmed by the IPL; nevertheless, it should be noted that this IPL, being reconfigurable and modular, could be used for a variety of systems, such as satellite control, deep-sea remote exploration, or industrial process control.

If the use of manipulators is ever to become more widespread than it currently is, then the ability to program these machines must become available to the researcher or worker who may not have a background knowledge in computers or robotics. By introducing an IPL into existing manipulator systems, floor-level workers will be able to run complex and critical routines, while the actual coding of lower-level tasks for these machines that the icons represent can be limited to professional programmers.

This paper is organized a s follows: The next Section describes previous efforts in this area and in Section 3, we briefly discusses the history of our IPL, and how we implemented a test version to demonstrate the effectiveness of such a scheme. Section 4 describes our current version of IPL called Onika and in Section 5, we discuss several research issues that we are pursuing. Finally, in Section 6, we summarize the paper.

2 Previous Work

An iconic programming language is distinct from what is known as a *visual programming language* (VPL), in that an IPL[1] relies on the user's association of actions and objects with pictures, shapes, and colors, rather than with less-informative flowcharts, textual information, and cryptic coding sequences. Iconic interfaces can be very useful for training new users to effectively operate their systems. Icons have proven to be easily identifiable and have simplified the tasks of moving through directories, accessing files, and running executables as in Macintosh and Windows environments. An IPL extends this idea by identifying an icon with a specific action or object; by combining icons in a certain way, an application can be described and executed.

A considerable amount of effort has been expended for developing graphical interfaces and visual languages to define an application. Visual programming has been applied to many diverse domains and an excellent review may be found in [11] . In this short review of previous work, our goal is to provide a perspective and motivation for the features that we chose to include in the development of Onika.

Hanne and Hoepelman suggest a "natural language" (NL) approach, where questions and statements are made based on simple English sentences[2] . The disadvantage of this is that "natural language" is only natural to those who communicate in an Indo-European language. For instance, the structure of Chinese and Japanese languages are quite different than that of English, and Amelsan (American Sign Language) structurally bears little resemblance to any spoken language. We feel that to be useful, an IPL should be conceptual, rather than English-oriented, and we have used this approach in our work. There are precedents for this approach; for instance, international traffic signs use concepts rather than written language. Besides, a visual robotic programming language using the NL approach would be so complex to parse as to minimize any advantage gained from using icons.

Leifer et. al. use icons to represent manipulation primitives [4]; these icons are then combined to construct a manipulation program. For the non-engineering type of person, programs can be easily created, debugged, and modified without a detailed knowledge of programming or computers in general. However, for the engineer who must decide whether or not to use PID control, force control, hybrid control, and so forth, the lexicon of the language of the interface must be expanded to include more than one level of programming. Furthermore, conditionals (such as if/then/else) should be added to create a robust vocabulary, and a grammar must be added so that the user is kept from creating impossible routines (e.g. *Move-To Move-To Ball Move-To Pick-Up*). Furthermore, the IPL should allow the creation of parallel-task applications, and the user should be able to completely redefine icons and their meanings without diminishing portability. In our research we are developing methods to implement these necessary programming concerns in a visual manner which is easily understood by the user.

Mahling and Croft introduce a visual programming language for the acquisition and display of plans for a planning system [5]. They have icons for acts and icons for relations and objects, which are used for creating plans to achieve some logical goal. What is noteworthy about their set-up is the notion of an expandable library of icons. Clearly, this is a beneficial thing to have. However, to minimize user confusion, only appropriate icons should be available to the user – for example, if a manipulator doesn't have a camera, then vision-routine icons should not available as icon choices for an application during its creation. Some method of organizing icons by category and domain is needed to prevent users from creating grammatically-correct but non-functional applications. A method of categorization of types and purposes of icons into some grammatical dichotomy is a major goal of our research.

Flow chart methods are often suggested, as Angelaccio et. al. have done for E-R oriented databases [1]. While flow charts lend themselves well to mapping the flow of a routine, they can look intimidating and are not as friendly to the user in terms of presentation of information as pictures would be. This is a much more important consideration than many people might think; user-friendliness improves productivity. Furthermore, arrows from one flow element to the next unnecessarily waste screen space. In the iconic approach we propose the creation of applications from job icons, the icons sit adjacent to each other; since icons do not require text to describe them, they can be much smaller than flow chart elements and yet convey the same amount of information and sense of program direction. This minimizes wasted space, and allows more routine elements to be seen at a time, which aids application development.

Flow chart methods are not at all useful when describing how a job is defined by servo tasks running simultaneously, and so

[1] Although *IPL* technically refers to the grammar, syntax, and manipulation of the programming elements of a particular type of human-machine interface, rather than the interface to the machine itself, we shall use both the terms *IPL* and Onika throughout this paper as reference to both the language and the interface.

when defining jobs from task we use a control-block form. Additionally, the pictures in the icons are more easily identifiable than the generic boxes of a flowchart with respect to specific routines. A flowchart element must represent a "complete" event (action + objects), since all flowchart elements of a type should theoretically be interchangeable. An IPL, on the other hand, could have an object easily replaced in an event without effecting its dependent action (and all of the values possibly associated with it), and vice-versa.

Mussio et. al. use icons which vary in shape, and are thus assembled in sort of a jigsaw puzzle ensuring correct grammar [7]; certain icons can only interlock with other icons (both left-right and up-down). Their icons resembled liver cells; a hepatologist was guided to create valid models of the liver and test them by using it. Glinert [12] describes a system called ProcBLOX that uses jigsaw puzzle pieces to present program constructs. Onika uses icons, as mentioned before, that are both shape and color coded puzzle pieces. In this context, a foremost concern in our research is exploring the best way for icons to be identified by class and grammar, whether by shape, color, size, or some other visual difference.

In the next section, we describe an initial version of our IPL called *Bookworm.*

3 A History of Our IPL

Our first prototype IPL called Bookworm was developed to demonstrate the effectiveness of iconic programming, and to discover some of the issues which would demand investigation in the research and development of a more powerful IPL. The Bookworm IPL was used to combine jobs and objects into applications, where the jobs were defined using code (in the newer IPL, Onika, the jobs used in the creation of applications are themselves iconically defined rather than textually defined).

Bookworm used a specific grammar to decide which icons may follow other icons in a story. For instance, an action icon which required an object (e.g. *Move-To* <object>) could not be followed by another action icon; it had to be followed by an object. Similarly, objects had to be preceded by an appropriate action-requiring-object icon. The grammar was immediately understandable to the user, since we used colors to identify icons: an icon whose right half is blue must be followed by an icon whose left half is blue, and so forth. There were three different types of icon "words" (green-green, green-blue, and blue-green, representing *action, action requiring object,* and *object,* respectively; we are currently performing research to discover exactly how many different classes of words are required for our latest IPL, Onika). The grammar of an icon was defined when the icon was created or modified, so that color did not need to be the identifying key; it could instead have been the shapes of the edges of the icons, for instance (useful if the user is color-blind).

Bookworm also conformed to the conventions of the platform on which it exists. For instance, the Macintosh version could be run from pull-down menus and command keys as well as icons; the Sun version of our present IPL, Onika, similarly uses SunView and XView conventions. In this way, users familiar with a particular platform are more easily be able to anticipate how the IPL reacts to commands and keystrokes.

Although Bookworm did not run a program on an actual manipulator, it did generate simulation files for use with ROBSIM, a NASA/Langley robotic simulator. When the "Simulate Story" icon was selected, AL code was generated through a four-pass method. First, Bookworm looked for holes in the story, and aborted the simulation if it found any. Next, Bookworm checked value panels for object locations, and defined variables for those locations, which it inserted into the AL file. On the next pass, those variables were initialized, and, finally, the meat of the code was produced.

Bookworm was strictly a higher-level IPL. At an early stage, we recognized that lower-level routines could also be coded using icons, although an entirely different grammar would be required, since lower-level routines are very specific and represent quite abstract concepts. Onika, the current IPL under development, is much more stratified than Bookworm and can be used to create lower-level jobs as well as higher-level applications, making itself more useful to programmers while still keeping lower-level details transparent to less technically-oriented users.

Onika's higher-level interface under development resembles Bookworm strongly but allows for a much expanded grammar (including conditionals, loops, and error-trap routines). By using icons and visual grammar cues to identify procedures, we avoid many of the problems of the flowchart approach to visual programming, including usage of screen space (much of which is wasted in flowchart methods), dependence on English, and problems in recognition of routines (one flow chart box looks pretty much like any other). Our icon "stories" are concise, compact, and easily read.

Onika's lower-level interface, designed to be used by experienced persons, resembles nothing so much as an IC CAD interface. Icons representing tasks, and having certain input and output pins, are combined into control block diagrams (connections are done automatically by the system, relieving the user of that tedious burden). Instead of having to change lines in pages of cryptic real-time code, the user can manipulate tasks graphically, retrieving and changing task information simply by clicking on the task's icon with a mouse. Activation and deactivation of one or all tasks is done with one keystroke, and task configurations can be modified as simply as selecting a task, deleting it, and dragging over another task to take its place.

Clearly, both the lower- and higher-level interfaces of Onika must rely on an underlying software support structure. Onika can be tailored for use on any system architecture, using any real-time system. The following section discusses Onika and its relationship to our own system architecture.

4 Onika's Position in the System Architecture

Figure 1 shows the system software architecture for reconfigurable sensor-based control systems[9][10] that we have developed. In this architecture, we have defined several types of

routines. The most general manipulator routine is the *application*, which specifies some goal to be achieved. For example, "Wash the dishes" is a fairly typical application in day-to-day life. An application is defined by some serial flow (or flows) of *jobs*, which define some singular action affecting an object. "Lift the plate," "Put scrubber on plate," and "Scrub" are good examples of jobs executed serially to achieve the application goal of washing the dishes. Finally, a job can be loosely described as being defined by a collection of *tasks*, all of which are running concurrently to fulfill the conditions of a job, and all of which have certain input and output functions. "Perform inverse dynamics," "Resolve acceleration of scrubbing unit," and "Invert scrubbing-arm jacobian" are all examples of tasks running concurrently to finish the job "Scrub." Tasks themselves are defined by various subroutine calls and are textually coded in *C* language.

Onika can be used to create both jobs from tasks *and* applications from jobs. Users can open a *task lexicon* (Figure 2), and drag icons representing tasks from the lexicon to a *job canvas* (these icons are simple CAD representations, and are created on the fly). Once on the canvas, a task is created on the underlying real-time system, and the icon representing that task automatically connects itself to other task icons based on the input/output variables of all the tasks in question (Figure 3). This allows the user to easily see whether or not a complete job has been created. The user can modify certain values associated with each task, such as the frequency, the priority, and whether or not the task is active or inactive. Task icons can be cut and repasted, although only one instance of each task can exist on the canvas. To create a job from tasks, the user need not know anything about the underlying real-time system, nor know much about computers, but he or she must have some knowledge about robotics.

However, once a job has been completely defined (shown in Figure 4), the entire job can be saved and linked to an pictorial icon (Figure 5). This pictorial icon can be saved for general use to a *job dictionary*, and dragged to an *application workspace* to become part of an application (Figure 6), which can also be saved and reloaded. When the application is run, and the job is encountered in a program flow, the tasks associated with that job are automatically loaded and activated based on the manipulator description chosen by the user. When the job is complete, its tasks are destroyed, and the program flow moves to the next job in the application. Jobs can cut, copied and pasted to the application as well, and of course multiple instances of jobs in the application are allowed. All of this is transparent to the high-level user. In order to create applications from jobs, the user need not know anything about the underlying support system, computers, or even anything about robotics. He or she simply needs to know how to drag job icons from one location of the screen to the other in some meaningful serial order ("Move here, then do this, then move there, then do that..."), the background grammar-checker ensuring that syntactically impossible applications cannot be generated.

Although not yet implemented, it is planned that applications will be used to define other applications (for instance, the application "Wash the dishes" could be a sub-application of the application "Do the housework"), and that applications could follow several flow branches (or even parallel flow branches) based on conditions at some point in the application.

It is expected that non-technically-oriented users would primarily create applications from jobs, and that the more robot/computer-oriented user would create the jobs from tasks. Experiments with the prototype IPL Bookworm have shown that most users can learn create usable applications from jobs in less than a half-hour.

While the implementation of an interface for technically-oriented users to define jobs from tasks is fairly straightforward, the implementation of powerful yet user-friendly interface for people having little or no background in robotics or computers to create applications from jobs is not. The next section discusses some of the research issues that we will be exploring while developing the IPL and supporting architecture.

5 Research and Development Issues

In order to develop the iconic human-machine interface for creating *applications* from *jobs*, several important research and development issues will need to be explored. These include (but are not limited to) the development of a grammar and the presentation of information to the user.

First, a grammar for the interface must be developed. This task, in turn, raises three other issues: first, how to represent a concept graphically and identifiably within a limited amount of screen space (the icon); second, how to classify the job icons as to grammatical types and identifying the number of types that will be needed, such as "self-contained action," "action requiring object," "object," "conditional," "modifier," etc. (the dichotomy); and third, how present visual grammatical clues to the user, so that the he or she does not waste time by attempting to create non-grammatical applications (the syntax).

In addition to the development of a grammar, the organization of information on the screen, and the devices by which it is affected, are also of paramount importance, and research will be performed to maximize their effectiveness. Interface controls to create, simulate, and run manipulator applications must be easily interpretable and used. The presentation of the application under development must allow the user to clearly see and understand the routine he or she is creating. Background error checking should immediately indicate when an impossible application is being built and prevent its occurrence, so that later debugging is kept to a minimum. The interface itself should conform to the established customs of the platform on which it exists (for instance, on the Macintosh, one would expect pull-down menus and command keys to perform operations similar to those that the user would expect they would in other Macintosh programs). Finally, all lower-level details must be transparent to the user; the interface should be developed with non-technically-oriented users in mind. These steps are necessary to create an interface which can be used at all times, and under any conditions, by users at any level of technical expertise.

We are especially interested in determining if an iconic programming interface will reduce the training time, experience,

and education necessary to operate machinery which at lower levels is quite complex. Informal tests have shown that users who have some familiarity with computers and window interfaces can learn to proficiently use the prototype IPL Bookworm in less than a half-hour. We plan on doing extensive tests on an expanded version to see how users who are inexperienced with using computers and/or robotics adapt to Onika.

6 Summary

We have described an iconic programming interface for creating applications for sensor-based systems such as manipulators. This IPL, called Onika, will allow a user to control both higher-level and lower-level routines. Lower-level details are kept transparent to non-technically-oriented users. With only a couple of hours of training, we expect that users will be able to construct complex applications for manipulators, despite any lack of previous experience with programming and/or robotics. To make the IPL as effective as possible, research is being performed to determine the proper grammar and visual presentation for the IPL.

Acknowledgments

Partial support for this research was provided by NASA, Sandia National Laboratories, the Department of Electrical and Computer Engineering, and The Robotics Institute, Carnegie Mellon University.

The authors would like to thank Philip Morris, Mike Goode, Jack Pennington, and all of the other researchers at NASA/ Langley's LTM facility for the use of machines and time to complete the Bookworm prototype IPL. Additional thanks go to Dr. Rich Volpe at JPL for his help in defining the task/job relationship.

More information on Onika, Chimera real-time operating system, and reconfigurable software can be obtained by contacting Professor Pradeep K. Khosla at Department of Electrical and Computer, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

References:

[1] Angelaccio, M., Catarci, T., and Santucci, G. "QBD*: A Fully Visual System for E-R Oriented Databases," 1989 IEEE Workshop on Visual Languages, Oct. 4-6, 1989, pp. 56-61, Rome, Italy.

[2] Hanne, K. H. and Hoepelman, J. Ph. "Combined Graphic- and Natural Language-Interaction (Design and Implementation)," Proceedings of Graphics Interface '88, June 6-8, 1988, pp. 105-111, Edmonton, Alberta.

[3] D.B. Stewart, D.E. Schmitz, and P.K. Khosla, "Implementing real-time robotic systems using Chimera II," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Cincinnati, OH, pp. 598-603, May 1990.Chimera

[4] Leifer, L., Van der Loos, M., and Lees, D. "Visual Language Programming: for robot command-control in unstructured environments," Proceedings of the Fifth International Conference on Advanced Robotics: Robots in Unstructured Environments, June 19-22, 1991, pp. 31-36, Pisa, Italy.

[5] Mahling, D. E. and Croft, W. B. "A Visual Language for the Acquisition and Display of Plans," 1989 IEEE Workshop on Visual Languages, Oct. 4-6, 1989, pp. 50-54, Rome, Italy.

[6] Miyao, J., Wakabayashi, S., Yoshida, N., and Ohtahara, Y. "Visualized and Modeless Programming Environment for Form Manipulation Language," 1989 IEEE Workshop on Visual Languages, Oct. 4-6, 1989, pp. 99-104, Rome, Italy.

[7] Mussio, P., Pietrogrande, M., Protti, M., Colombo, F., Finadri, M., and Gentini, P. "Visual Programming in a Visual Environment for Liver Simulation Studies," 1990 IEEE Workshop on Visual Languages, Oct. 4-6, 1990, pp. 29-35, Skokie, Illinois.

[8] Prusinkiewicz, P. and Knelsen, C. "Virtual Control Panels," Proceedings of Graphics Interface '88, June 6-8, 1988, pp. 185-191, Edmonton, Alberta.

[9] Stewart, D. B. "Real-Time Software Design and Analysis of Reconfigurable Advanced Sensor-Based Systems," Ph.D. prospectus, The Robotics Institute, Carnegie Mellon University, March 31, 1992.

[10] Stewart, D. B., Volpe, R. A., and Khosla, P. K. "Integration of software modules for reconfigurable sensor-based control systems," in *Proc. 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS '92), Raleigh, North Carolina, July 1992.

[11] Myers, B. A., Taxonomies of Visual Programming and Program Visualization, Journal of Visual Languages and Computing, 1990, pp 97-123.

[12] Glinert, E. P., Out of Flatland: Towards 3-D Visual Programming, Proceedings of 1987 Fall Joint Computer Conference, October, 1987, Dallas, TX. pp 292-299.

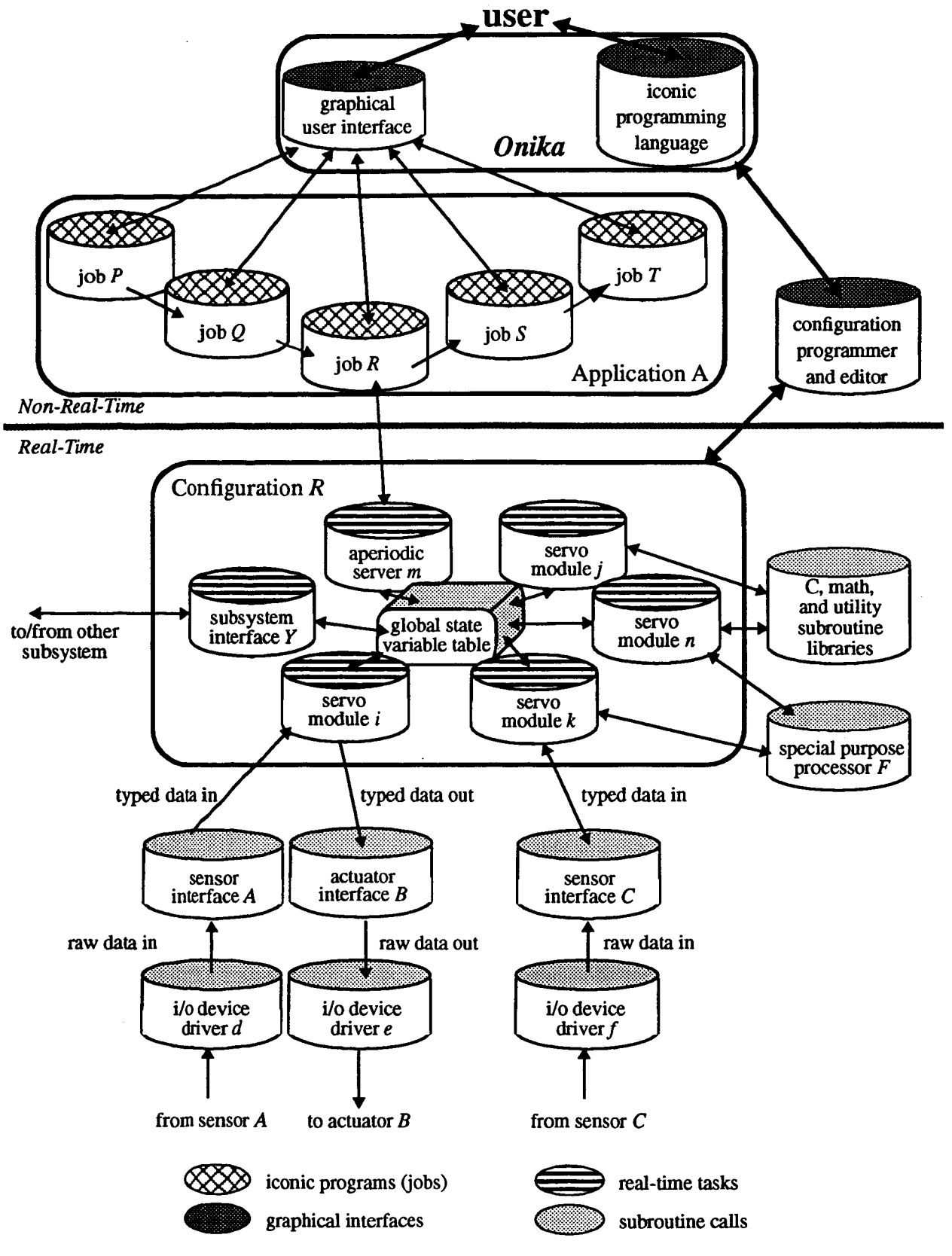


Figure 1: Software Architecture for Reconfigurable Systems

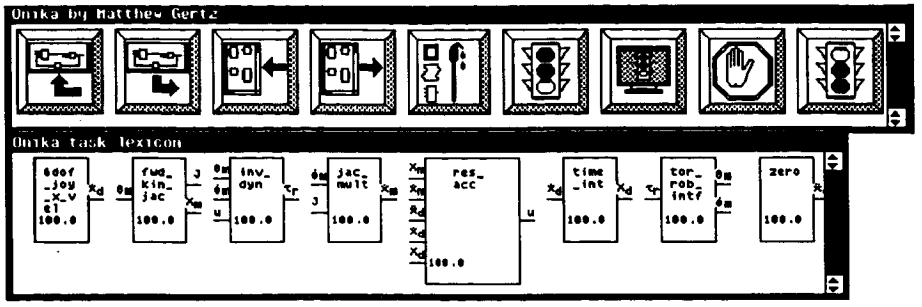


Figure 2: Loading the Task Lexicon into Onika (Sun version of Onika)

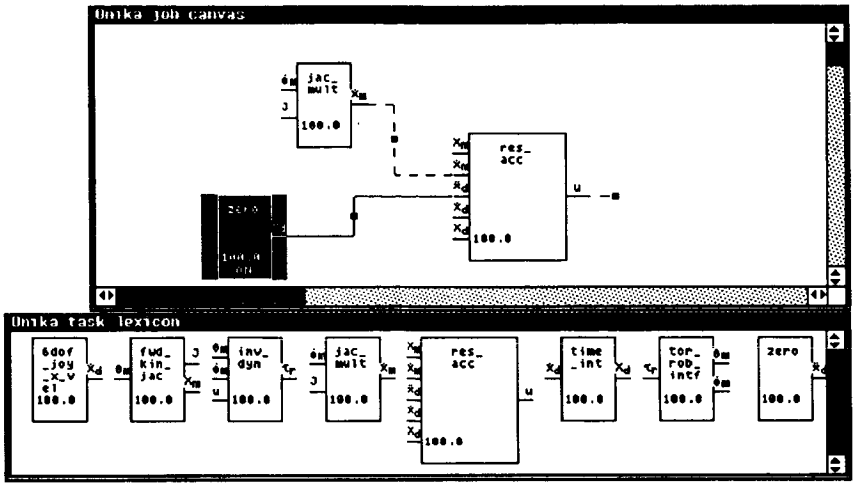


Figure 3: Pulling task icons onto the job canvas (Sun version of Onika)

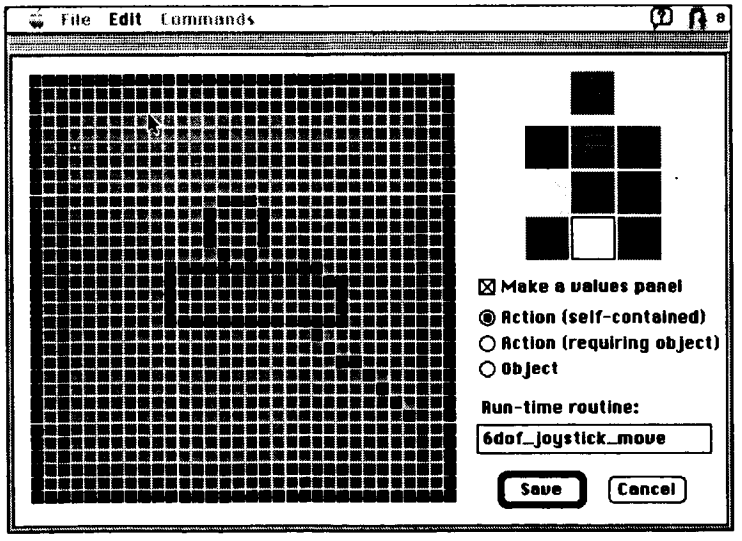


Figure 5: Creating an Icon for the Job (Mac II version of Bookworm)

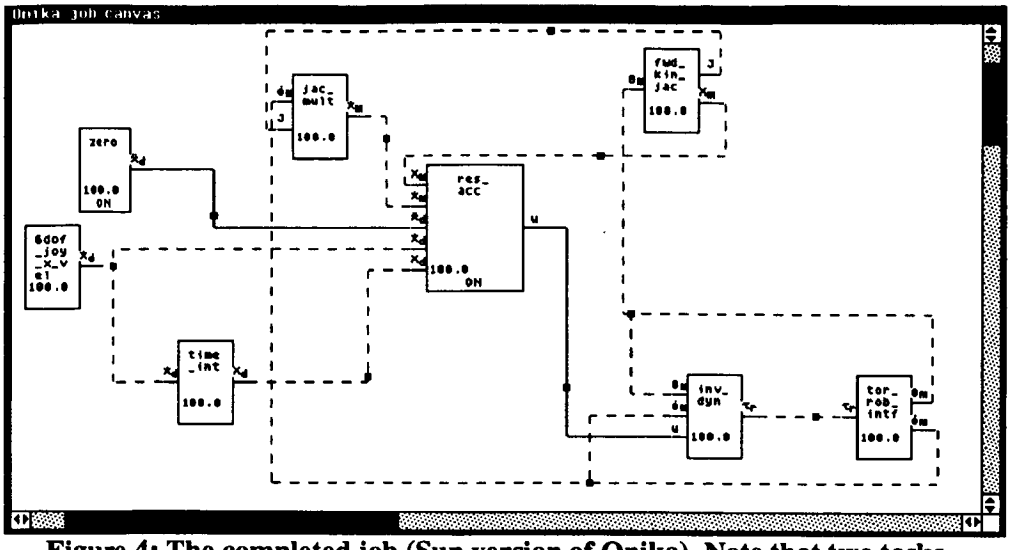


Figure 4: The completed job (Sun version of Onika). Note that two tasks are on and generating output as denoted by solid lines

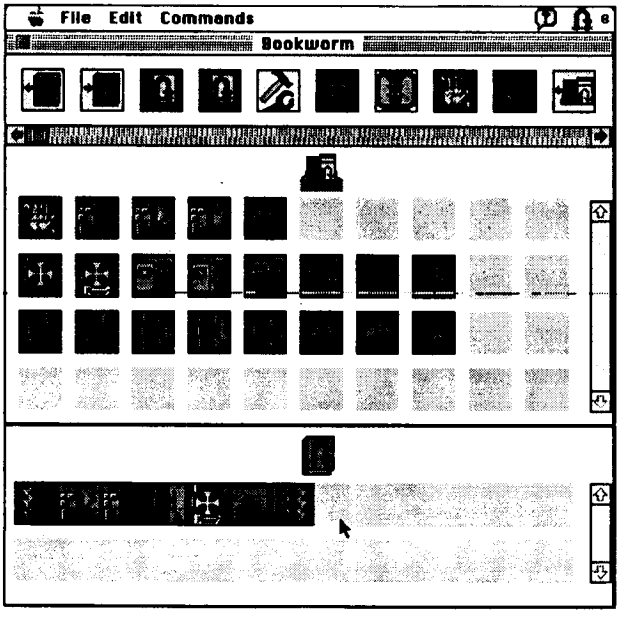


Figure 6: Using the Job's Icon in an Application (Mac II version of Bookworm)

N93-32099

REMOTE SURFACE INSPECTION SYSTEM

S. Hayati, J. Balaram, H. Seraji, W. S. Kim, K. Tso*, V. Prasad**
Jet Propulsion Laboratory
California Institute of Technology
*SoHar Corporation
** California Institute of Technology

Abstract

This paper reports on an on-going research and development effort in remote surface inspection of space platforms such as the Space Station Freedom. It describes the space environment and identifies the types of damage for which to search. This paper provides an overview of the Remote Surface Inspection System that has been developed to conduct proof-of-concept demonstrations and to perform experiments in a laboratory environment. Specifically, the paper describes three technology areas: 1) manipulator control for sensor placement, 2) automated non-contact inspection to detect and classify flaws, and 3) operator interface to command the system interactively and receive raw or processed sensor data. Initial findings for the automated and human visual inspection tests are reported.

Introduction

Space platforms such as the Space Station Freedom (SSF) are complex and expensive scientific facilities which must operate in *harsh* and *remote* environments. Such facilities must be maintained, i.e., inspected, replenished and repaired periodically to assure safe operations for the crew and to provide reliable experiments and apparatus for scientists.

Although inspections and repairs may be performed by astronauts, there is considerable risk and cost in Extra Vehicular Activities (EVA). Studies have shown that astronaut EVA time is over-subscribed for SSF maintenance and therefore whenever possible tasks should be performed by means other than EVA. In addition, the Freedom will be unattended 87% of the time for the first few years. This precludes any EVA activity except during Shuttle visits, when experiment tending and scientific research have greatest emphasis. Given the state of robotics technology, non-contact inspection leading to preventive maintenance, as opposed to breakdown repair, presents a cost-effective opportunity for the use of robotics/teleoperation in space. The recent "Space Station External Maintenance Task Team" report identified inspection as the most tedious and time-consuming task that needs automation [1].

Due to the SSF's large size, it is not logistically practical to mount fixed observation cameras and other sensors to cover the entire facility. Telerobotics is therefore a natural technology for remote visual inspection of the Space Station and other large space platforms.

This paper describes the research and preliminary results on remote surface inspection and provides a brief review of the newly developed Remote Surface Inspection system at the Jet Propulsion Laboratory. The organization of this paper is as follows: Section 1 provides a brief background on the space environment, including highlights from the Long Duration Exposure Facility (LDEF). Section 2 furnishes information on the Space Station Freedom and identifies inspection tasks required for its maintenance. The Remote Surface Inspection system is described in Section 3 where its architecture and capabilities are outlined. Section 4 describes our current approach to both human visual inspection and automated

inspection. In Section 5, some issues in machine-vision-based inspection are discussed. Experimental results and conclusions are presented in Sections 6 and 7, respectively.

1. SPACE ENVIRONMENTAL EFFECTS

The low earth orbit environment consists of many hazardous elements that may lead to a degradation of material properties and threaten the surface elements of satellites and space platforms. Many factors such as exposure to mono-atomic oxygen, solar wind, ultraviolet rays, radiation reflected from the earth, thermal cycling effects due to the motion in and out of the earth's shadow, meteoroids, and space debris will cause damage to varying degrees. Past space missions and in particular the Long Duration Exposure Facility (LDEF) have shown that severe damage and degradation can result to exposed and protected flight hardware surfaces, particularly to organic materials, thin foils, and certain other coatings, when the exposure is for an extended period of time [2].

The LDEF spacecraft was a 30 foot long, 14 foot diameter, 14-faced (12 sides and two ends), open-grid structure on which a series of 86 rectangular trays were used to mount experiment hardware. The spacecraft was exposed to the on-orbit space environment for about 6 years, through a 257 kilometer altitude orbit, approximately the same orbit band that Space Station Freedom will utilize. The Meteoroid and Debris Special Investigation Group (M&D SIG) found more than 34,000 impact features on all space-exposed surfaces, most below 1.0 mm in diameter. Figure 1 shows several samples of LDEF material. Table 1 summarizes the types of features by size. The largest impact crater was 5.7 mm in diameter [3, p. 5] and probably caused by an object about one millimeter in diameter. The smallest crater identified to date [3] is 0.1 micrometers in diameter.

Additionally a wealth of information pertaining to the constantly changing space environment and its impact on long duration spacecraft surfaces has been analyzed and reported [4]. Information directly relevant to design of telerobotic spacecraft surface inspection systems include: (1) how surface orientation (Earth facing, Space Facing, Leading Edge, Trailing Edge) alters the rate of surface impact or damage; (2) the flux rates (number-of-impacts/area/second) associated with man-made (debris) and natural (micrometeoroid) impactors; (3) the rates at which space debris and meteoroids are increasing in Low Earth Orbit; (4) the morphological features of surface damage/impacts as a function of the physical form and composition of both the impactor and the impacted surface.

On the LDEF satellite, the leading edge was mostly impacted by debris (or man-made impactors), while the trailing edge was mostly impacted by meteoroids (natural) [3]. A short-term increase in micro-particle debris impact (or flux-rate increase) following LDEF deployment has been observed and attributed to increased Shuttle activity during this period. On average, the impacts in the first year were higher that in the following years.

The LDEF post-flight investigation is not yet complete. For current information and results, the reader is referred to the Environmental Effects newsletter, published by the LDEF Systems Special Investigation Group [5].

Figure 1 (a) This pre-deintegration view shows one module of LDEF tray H12 with the thermal blanket partially peeled back exposing the detector stack below.

Figure 1 (b) A penetration hole through one of the H12 multi-layer blankets.

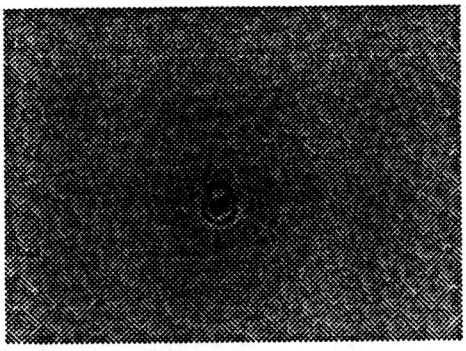


Figure 1 (c) A concentrically-ringed impact feature into white-painted aluminum surface.

The present average rate of increase of space impactors (debris and micrometeoroids) is around 5%/year. However, the LDEF information obtained so far has validated existing on-orbit micro-meteoroid and orbital debris (MMOD) models, which predict an approximately 11% yearly increase in particulate contamination of the Earth-orbit region [6]. It is estimated that this number will increase to about 20%/year by the turn of this century [7]. These results have prompted design rules for Space Station that include avoiding use of organics and thin-film foils for the SSF external surfaces, in favor of anodized aluminum and Z93 coatings (flat black or flat white, as thermally appropriate). External structure is being designed to withstand the average on-orbit effects for thirty years. These are reasonable countermeasures to naturally-induced, detectable wear.

Such information as the models provide can not only be utilized to design future spacecraft, but can also be used to derive requirements for an inspection system. For example, the micrometeoroid impact features shown in Table 1 strongly indicate that in order to use an inspection system to revalidate future MMOD models, the system must be capable of detecting very small flaws in the range of 0.2 to 6.0 mm on surfaces of varying shape and specularity under orbital lighting conditions. This must take place while satisfying safe clearance requirements, as well as other requirements for mobility and safety such as smooth motion, collision free scanning, and so on.

Table 1. Summary of LDEF Impact Features [2].

Feature Size (diameter)	Clamps,Bolts, & Shims	Tray Flanges	Experimental Surfaces	Totals*
< 0.3 mm	-	-	2911	3069
> 0.3 mm	-	-	763	763
< 0.5 mm	1318	1923	19342	27385
> 0.5 mm	161	419	2539	3119
Totals	**1479**	**2342**	**25555**	**34336**

* Note: the "Total" is greater than the sum of the individual column entries for the "<0.3 mm" and the "<0.5 mm" rows because some of the features contributing to the total were detected on intermediate surfaces such as between the tray flanges and the experimental surfaces.

2. SPACE STATION FREEDOM

The Space Station Freedom (SSF) is a large space platform with complex mechanical, electrical, thermal, fluid and gas interfaces, and a changing suite of internal and external scientific experiment apparatus. Over a 30-year design lifetime, Freedom will be adapted and upgraded as our knowledge of the effects of prolonged microgravity exposure on living creatures and inanimate objects advances, and we identify new experiments and processes to perform. On-orbit maintenance of such a complex, changing facility requires periodic as well as "on demand" inspection capabilities. Although subjective "eyes-on" observations during planned crew-EVA will gather much important data, telerobotic inspection offers precise repetition of calibrated sensor placement and positioning, enhanced (non-visible light) sensing, digital scene recording and matching, and greater automation in flaw detection and categorization.

Periodic inspection is required to ensure that potential problems are detected early on and changes in SSF external configuration and appearance are monitored. This type of inspection can be scheduled to take place *non-invasively*, e.g., when no other major activity is planned. On-demand inspections, for example in preparation for crew-EVA, can aid operations planning by assessing the condition of external SSF structure or interfaces at an EVA worksite. Revisiting previous worksites where work was suspended can determine if equipment left there is in order. Orbital Replacement Unit (ORU) installation site inspections can determine empty interface conditions, affecting tool manifesting for the next EVA visit.

Although at the present time the ground-based control of robotic devices is not part of the SSF baseline design, NASA is interested in performing a feasibility study to determine if future ground-based telerobotic operations can supplement on-board operations. In the several years prior to permanent manned operations, when the station will be mostly untended, ground-remote telerobotic operations could support both periodic and on-demand (in response to anomalies detected by dedicated sensors) non-contact inspections. These activities could provide useful, detailed preliminary information vital to planners who must make the most efficient use of limited crew-EVA during man-tended operations.

2.1 Inspection Requirements

Although at the present time, detailed inspection requirements have not been specified, NASA has emphasized the need for inspection in various documents and forums. For example, the Space Station Freedom External Maintenance Task Team Final Report (See [1], Appendix E) specifies high-level requirements for the inspection of the station and states that telerobotics should be utilized to accomplish some of the inspection tasks, in particular, routine and repetitive ones. A number of candidate tasks have been identified based on our interactions with engineers at the Johnson Space Center (JSC) and various scientists working on LDEF. These include inspection of (1) truss strut damaged by micrometeoroids, (2) cracks in structures, (3) shield area damaged by micrometeoroids, (4) thermal blankets, radiators, or solar panels damaged by micrometeoroids and atomic oxygen, (5) thermal/mechanical interfaces at ORU installation sites, (6) deployable mechanisms for incorrectly positioned latches, connectors, and other mechanical devices, (7) the SSF shuttle docking port before each docking, (8) damaged fluid and power lines in a utility tray , (9) effects of fluid leaks on optics, and (10) magnetic fields, plasma fields, and contaminant levels, especially hydrazine concentration.

3. TELEROBOTICS INSPECTION SYSTEM

In this section we describe a telerobotics inspection system that has been developed to perform human visual inspection experiments in realistic space-like environments and to develop and integrate new supervisory robotic and automated inspection techniques. A strong emphasis has been placed on duplicating critical space environment effects within the available laboratory space and budget. This system consists of local and remote sites. In our terminology, a local site is where the operator resides, which can be the habitation module or a ground station on the earth; the remote site is the task location where the robot is and the inspection takes place.

The remote site elements consist of one manipulator arm mounted on a mobile platform, which carries lights, cameras and other sensors for inspection and manipulation, and computers for processing real-time inspection data. Figure 2 shows the remote site of the system. The inspection task mockup board is a one-third scale of a section of the Space Station truss structure. Two tank ORUs are mounted on this truss structure to provide typical surfaces for inspection experiments. The manipulator is a seven-degree-of-freedom (7-DoF) Robotics Research Corporation™ (RRC) arm which is mounted on a one-degree-of-freedom motion platform. Since six DoF are sufficient for arbitrary placement and orientation of the end-effector, the "extra" DoF in the arm is used to provide direct control of the elbow position *independent* of the end-effector position and orientation. This is accomplished by using the "configuration control" methodology [8] developed at JPL to control the "arm angle," which is the angle between the arm plane and a reference plane. The control scheme provides

robustness to kinematic singularities and allows the user to specify weighting factors for the task requirements. The configuration control of the 7-DoF RRC arm with elbow positioning capability has been implemented for real-time control of the arm.

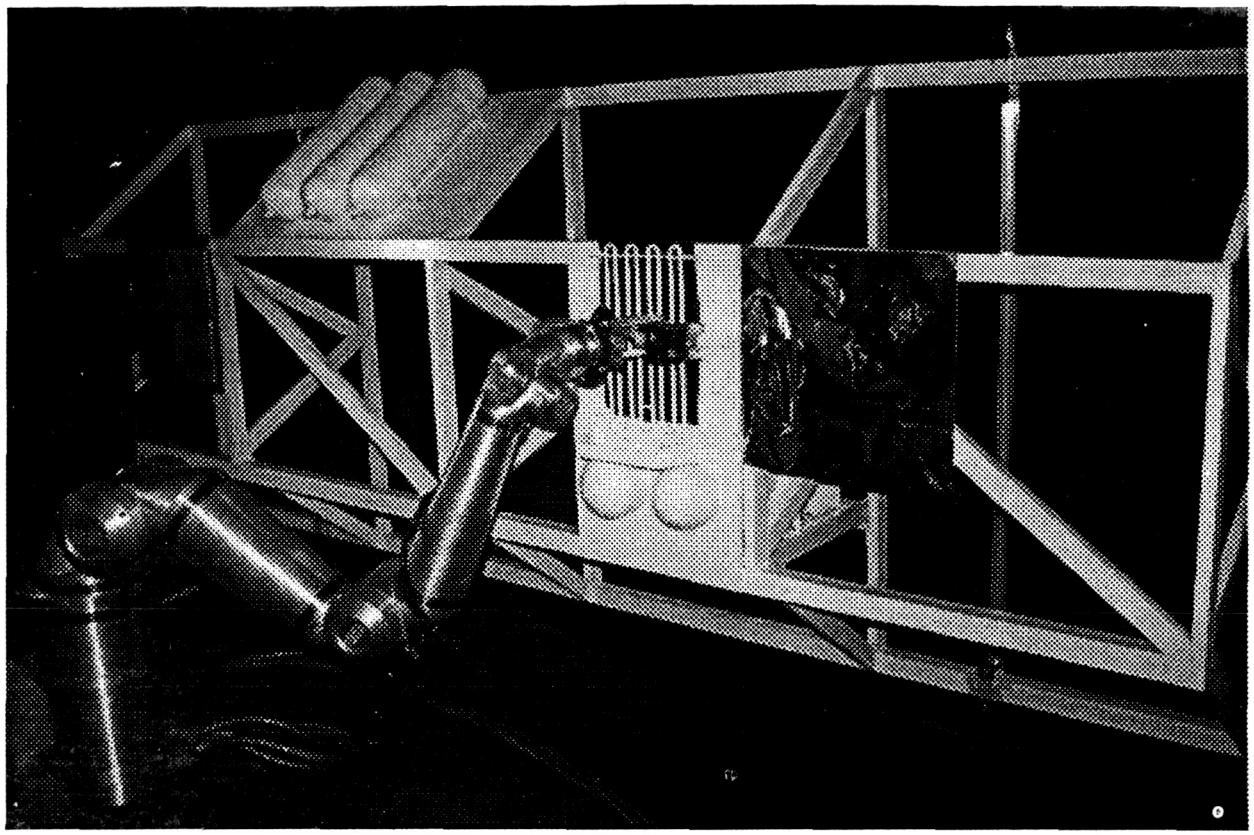


Figure 2. Remote Surface Inspection System, showing 1/3 Scale Space Station Truss Mockup, ORUs and Inspection Manipulator Arm.

The one degree-of-freedom platform provides base mobility to the RR arm and increases the workspace of the arm considerably. The mobile platform is treated as the eighth joint of the manipulator system. Following the configuration control approach, an additional (8th) task is defined to resolve this redundancy. The additional task is formulated as the control of the "elbow angle," which is the angle between the upper-arm and the forearm. When this angle indicates that the arm is over-stretched or under-stretched, the platform is moved automatically to restore the optimal condition, without perturbing the end-effector position and orientation. Thus the automatic motion of the platform prevents undesirable over-stretched or under-stretched arm configurations.

Reference [9] provides details of the architecture, algorithms, and hardware description of the manipulator control system. The arm functionally simulates the Special Purpose Dexterous Manipulator (SPDM) and the motion platform provides translational capability which simulates the mobility of the SPDM provided by the Space Station Remote Manipulator System (SSRMS), in a limited sense. Realistic SSF environmental effects are provided to improve the operator's perception in executing arm functions.

Lighting and viewing at the remote site is achieved by means of a controlled environment, i.e., black ceiling, floor, and curtains. Two sets of lights are used in the laboratory. One simulates Low Earth Orbit sunlight and the other simulates the controlled environment lighting of the SSF. Figure 3 shows the 1200W, 5600∞K +/- 400∞K, adjustable-focus Luxarc 1200 lamp for Sun-like illumination producing high contrast between shadowed and lit surfaces. This lamp is mounted on a moving platform with computer-controlled pan/tilt mechanisms and intensity to mimic realistic analog changes in sun angle and strength as the SSF orbits the earth. Two controlled lights are mounted on the end-point of the manipulator arm to provide close-up illumination and to light enclosed regions. In addition, three other cameras with pan, tilt, zoom, and focus control capability are mounted in the laboratory to provide functions similar to those that will be available on the SSF.

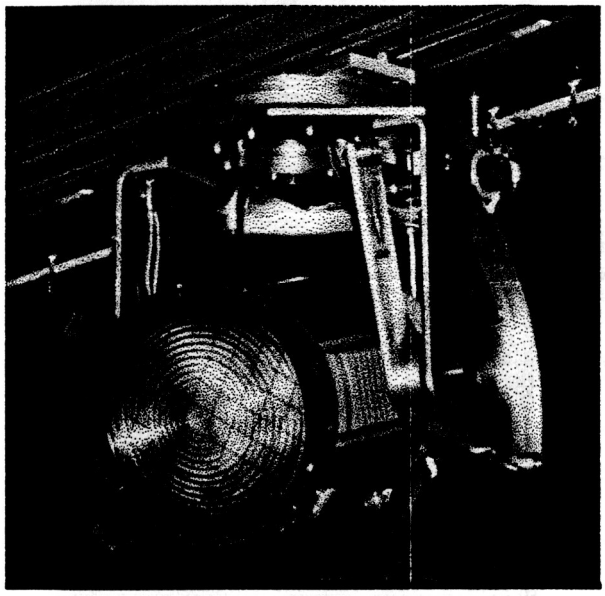


Figure 3. Computer-controlled, simulated solar illumination of the worksite is provided by this 1200W lamp with variable pan, tilt, focus and X-travel capability.

The local site provides operator interface hardware and software as well as a data logging/viewing and simulation facility. The control station is composed of three high resolution color monitors, a Silicon Graphics IRIS workstation, two shuttle-like joysticks, and a control panel that provides the camera, light, and video switch interface to the operator. Figure 4 shows the overall operator control station, housed in a Space Station cupola mockup that realistically simulates the equipment and operator space limitations. Figure 5 shows a block diagram of the Remote Surface Inspection System

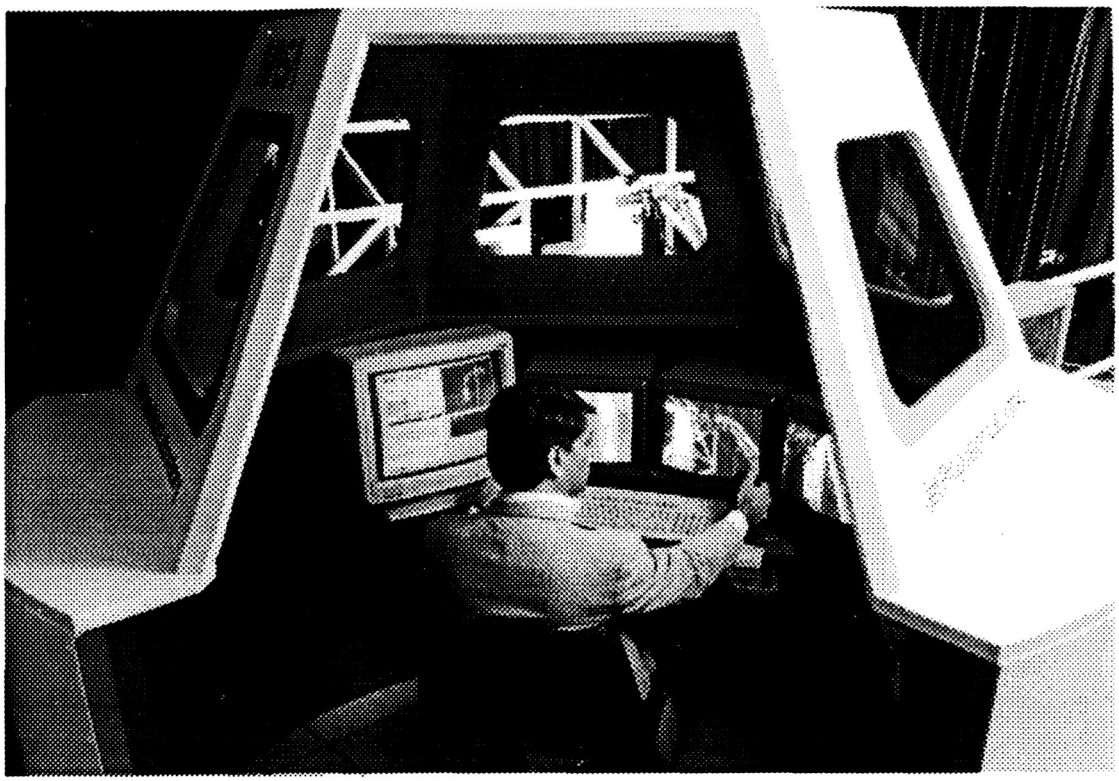


Figure 4. Remote Surface Inspection Laboratory, showing the Operator Control Station housed in a full-size cupola mockup of the Space Station.

4. INSPECTION STRATEGIES

Three complementary inspection strategies have been implemented. They are: 1) teleoperated human visual inspection, 2) automated scanning with human visual inspection, and 3) automated scanning with machine-vision inspection. The simplest and most reliable remote inspection technique is to present images of the area of interest directly to an operator and record those regions which the operator determines contain flaws. This technique relies on the operator to control the arm, lights, and cameras, in addition to performing visual inspection. In many instances, the operator's work load may be reduced by providing him with additional software tools such as automated scanning of the desired region, machine-vision inspection, and on-line flaw marking and annotation facilities. In the following, we will describe these inspection strategies in more detail.

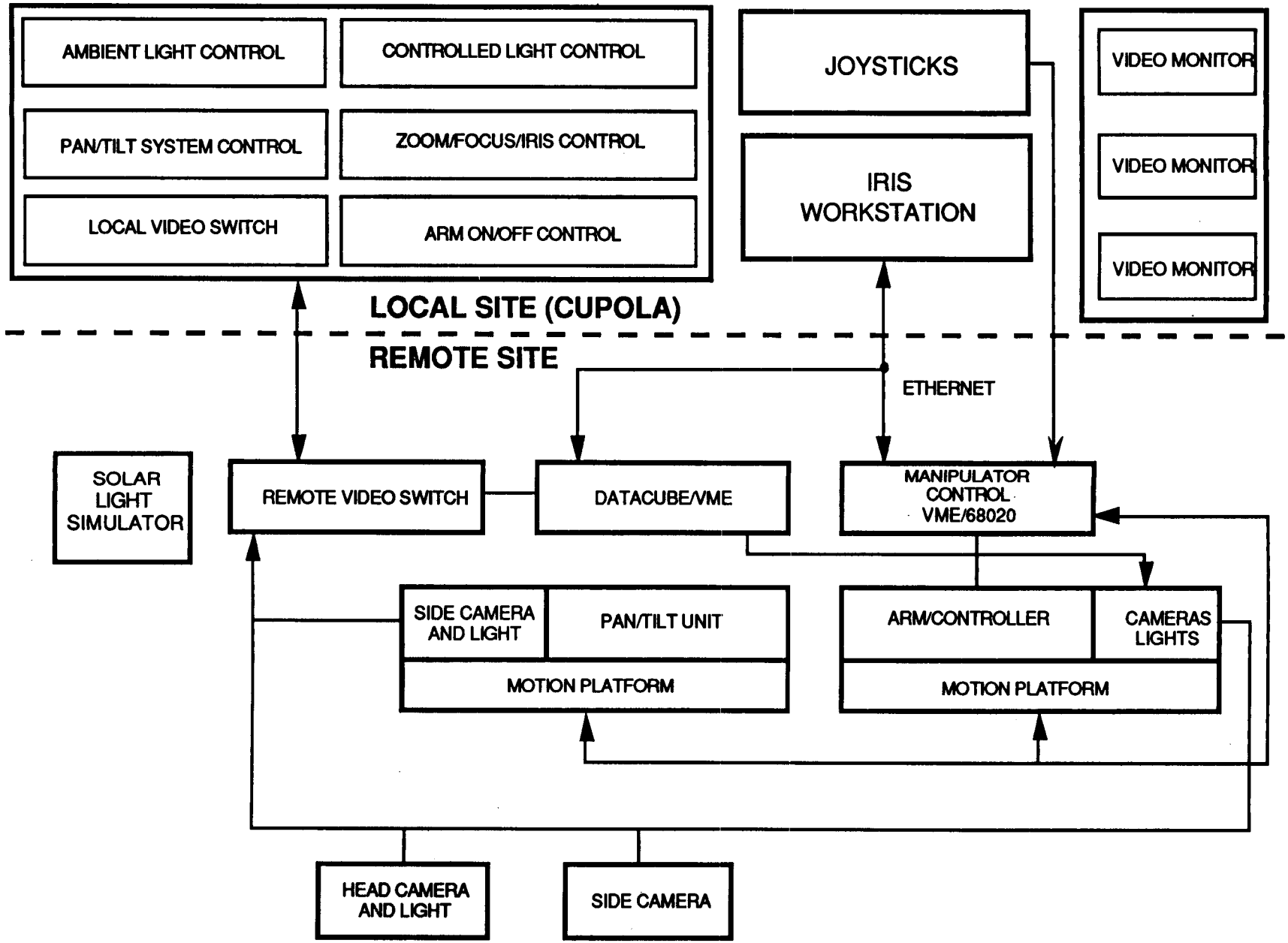


Figure 5. Hardware Block Diagram of the Remote Surface Inspection System

4.1 Teleoperated Human Visual Inspection

With human visual inspection, the operator inspects the surface through the camera monitors. If a flaw appears, he can stop the scanning and capture the close-up image of the object being viewed by the inspection camera and display it on the Close View window. The operator can then further examine the flaw, compare it with the ones marked in previous scans, and mark the flaw on-line, as will be discussed in more detail in the following subsection.

In this mode of operation the operator interacts with the Graphical User Interface (GUI, shown in Figure 6) on the IRIS to set the appropriate mode of teleoperation and uses two joysticks to move the arm. The interface also allows the operator to move the arm by specifying the target position in the operator-commanded auto-move mode. One important feature of this system is that the operator can control the arm in shared control mode, which means that he can easily modify the preprogrammed or automated motions to avoid obstacles or to slow down the motion and prolong the inspection. Since the arm has seven degrees of freedom, a trigger on one of the joysticks can be used to control the elbow rotation to better position the arm and avoid obstacles.

In auto-move mode the motion commands are generated by recording the arm positions in teleoperation and are stored in auto-sequencing scripts. These scripts can contain commands to perform other tasks such as image processing operations. The GUI also provides real-time graphical animation of the arm so that the operator can "view" the arm and its environment from directions not available using the actual camera. This capability is particularly useful when there is no direct viewing of the scene, as is the case for ground-based operations and for many on-board operations as well. Another window of the GUI displays digitized images that can be captured by any of the five cameras of the system. The operator can use either the camera monitors or the digitized images to inspect the object surface.

4.2 Automated Scanning

Object scanning can be made more efficient and reliable by an auto-scan planner. This allows the operator to simply designate an object or a region for which the system then automatically generates a scan path. Auto-scan planning includes the Far View, Close View and Object Definition windows shown in Figure 7. The Far View window displays the digitized image of the object being scanned. The image is captured from a camera at a great enough distance as to contain the entire object within the window. This window allows the operator to see the context of the area being inspected, while providing a means for mapping the object image to the actual object positions.

Figure 6. Graphics user interface to Remote Inspection System.

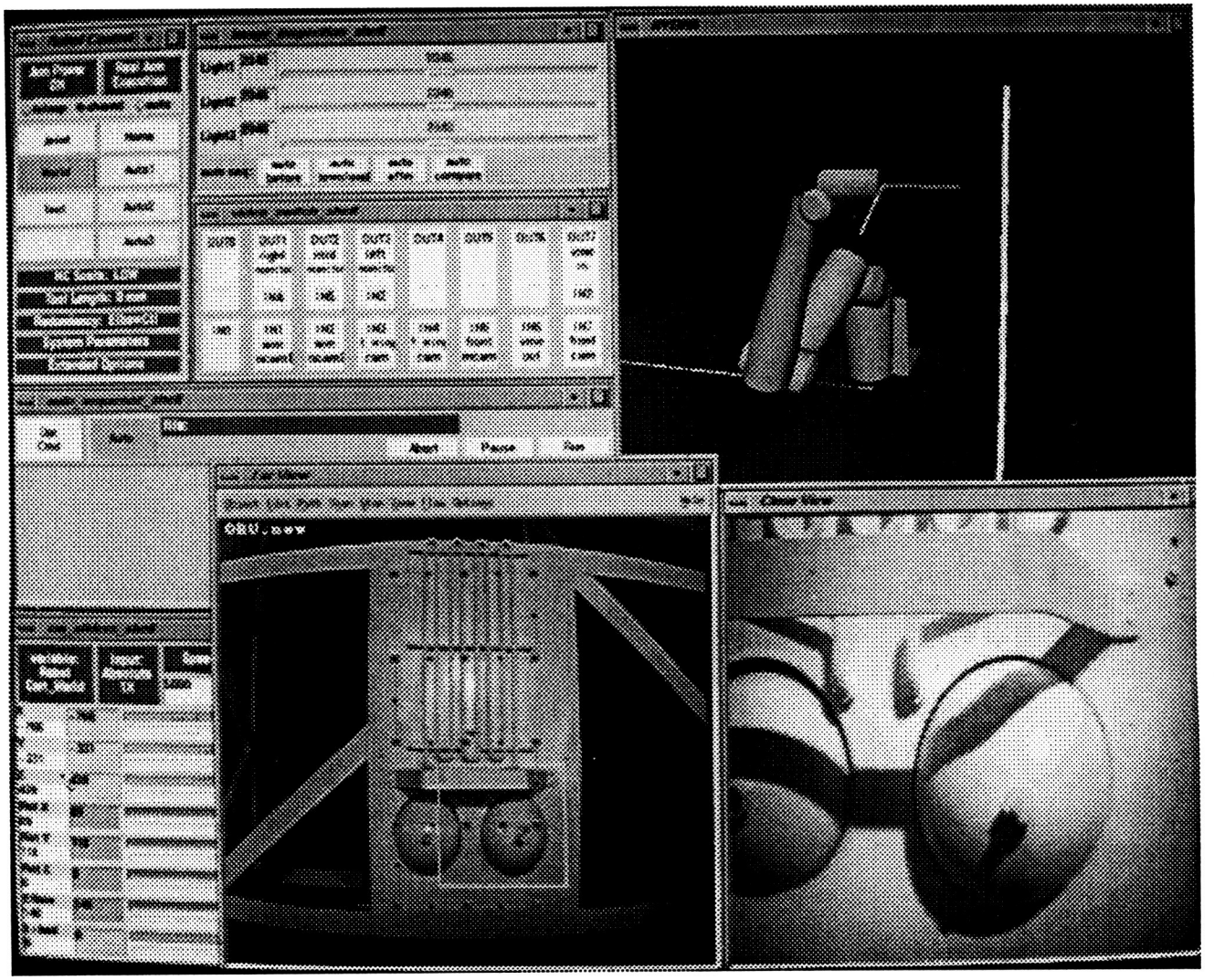


Figure 7. Automated Scanning Window.

Object designation is achieved by moving the camera to each corner of the object with the desired inspection distance and registering each corner by reading the arm position and marking the corresponding corner on the image. If no image of the object is available, the Far View window shows a wire frame polygon with the corresponding corners. After mapping, a scan path can be generated which covers the whole area with overlap in each swipe. Two default scan paths can be chosen by the operator: the *horizontal* path which scans the object horizontally from top to bottom, and the *vertical* path which scans the object vertically from left to right. Additionally, *via points* where the arm will pass through, and *vista points* where inspection will be made, are generated along the path. The vista points are placed to ensure that the inspection will cover the whole object. More via points are put along the path to ensure smooth and accurate scanning. Moreover, the operator can use the menu to insert, delete, and relocate via points or vista points as needed for scanning objects with irregular shapes. Figure 7 shows a horizontal scan path with a point relocated. The object mapping and scan path generation are needed only when a new object is introduced to the system. The data are saved for subsequent inspection and can be loaded by selecting from a list of objects using the menu. The operator can then initiate scanning from the current arm position or any via points on the path, scanning either forward or backward along the path. He/she can also use the menu to request that the arm move to designated via points.

4.3 Automated Scanning With Machine-Vision Inspection

Before flaws can be detected automatically, a *reference scan* of the entire object is obtained. Flaw detection is achieved by comparing the images of the subsequent *comparison scans* against the ones from the reference scan. The operator selects the type of scan from the menu. The scan type and the list of vista points are sent to the Inspection Subsystem which performs automated inspection as described in the next section. The Inspection Subsystem signals the auto-scan planner when a flaw is detected. Normally, the arm is stopped and the operator marks the flaw just as he would with human visual inspection. The operator can also ignore the signal if he determines from the monitor that it is a false alarm.

4.4 On-Line Flaw Manipulation

In addition to automated scanning and inspection, the integrated environment provides on-line flaw manipulation for remote surface inspection. It allows the operator to input, save, retrieve, view, and compare the location, image, and annotation of flaws from previous and current scans. The operator marks a flaw by first placing the cursor on top of it in the Close View window and then using the menu to mark it. The program saves the flaw location, extracts its image to a file, and allows to operator to enter an annotation. Flaw marking and interpretation can be greatly enhanced by the ability to review the flaws from previous scans. This information is summarized in the scan history table. Each column, labeled with the date of the scan, contains the flaws marked in that scan. Each row, labeled by a flaw ID, contains the same flaws marked in different scans. Each entry on the table shows the location of the flaw. The operator can view a previous scan by double-clicking on the date of that scan. All the flaws marked in that scan are then shown on the Far View window in Figure 7. This allows him to determine if there are any flaws that are missing or newly added in the current scan. Using a cursor control device such a mouse or trackball, the operator can double-click on an entry to display the image of that flaw. He can also double-click on the flaw ID to

display all of the flaws with that ID. This allows him to see the changes of a particular flaw in each scan. The annotation of a flaw can be brought up by double-clicking its image.

5. Automated Inspection

Automated inspection is presently used in industry to inspect printed circuit boards, mechanical components, and other specific and well defined objects [10, 11, 12]. The objective of our research is to develop automated inspection techniques that can perform inspection for any general surface without adapting the algorithms for these objects. In the case of the SSF, we are interested in providing the operator with an automated inspection system which could be used for various ORUs, truss struts, pipes and extended surfaces such as radiators. Our initial approach is to survey the entire SSF by using a manipulator arm to collect images and other relevant data. Assuming a certain level of repeatability of the manipulator arm, it will be possible to re-survey the desired locations after a period of time and then compare the two sets of images. Whenever the system comes up with a large discrepancy between the 'before' and 'after' images, it notifies the operator to confirm and log the flaw in a database. This simple image differencing technique provides a powerful inspection tool for the operator who only has to command the system and does not have to continuously visually inspect the images. This also provides an audible trace of previous inspection runs and findings. For this technique to work reliably, however, several technical issues must be resolved.

First, any differences in the ambient light for the before and after images will introduce large discrepancies. Normally, these differences are large enough that a simple differencing technique always indicates possible flaws on the surface. The second problem is that of registration accuracy between before and after images. Even small errors in the position and orientation of cameras produces large discrepancies, particularly in high contrast areas of the images such as edges. The third problem is due to erosion of surfaces due to atomic Oxygen and ultraviolet exposure to the degree that changes in the overall reflectivity of the surface produce false triggers by the inspection system. In the following we will describe our approach to deal with the first two problems and discuss initial results.

5.1 Ambient Light Removal

Controlled lights mounted on the arm end-effector, base, and the fixed cameras can be used to cancel the ambient light effects. The concept is based on differencing two images of the same surface; one taken with the controlled lights off and the other with the lights on. The resulting image is an artificial image of the scene as if there were no ambient light. This image is stored as a "before image." Similar operations are performed for the "after image" and only then the before and after images are differenced. Figure 8 shows the result of an experiment for flaw detection under a variable ambient lighting condition. Note that in this scene, it is not easy to detect the missing screw even by human visual inspection.

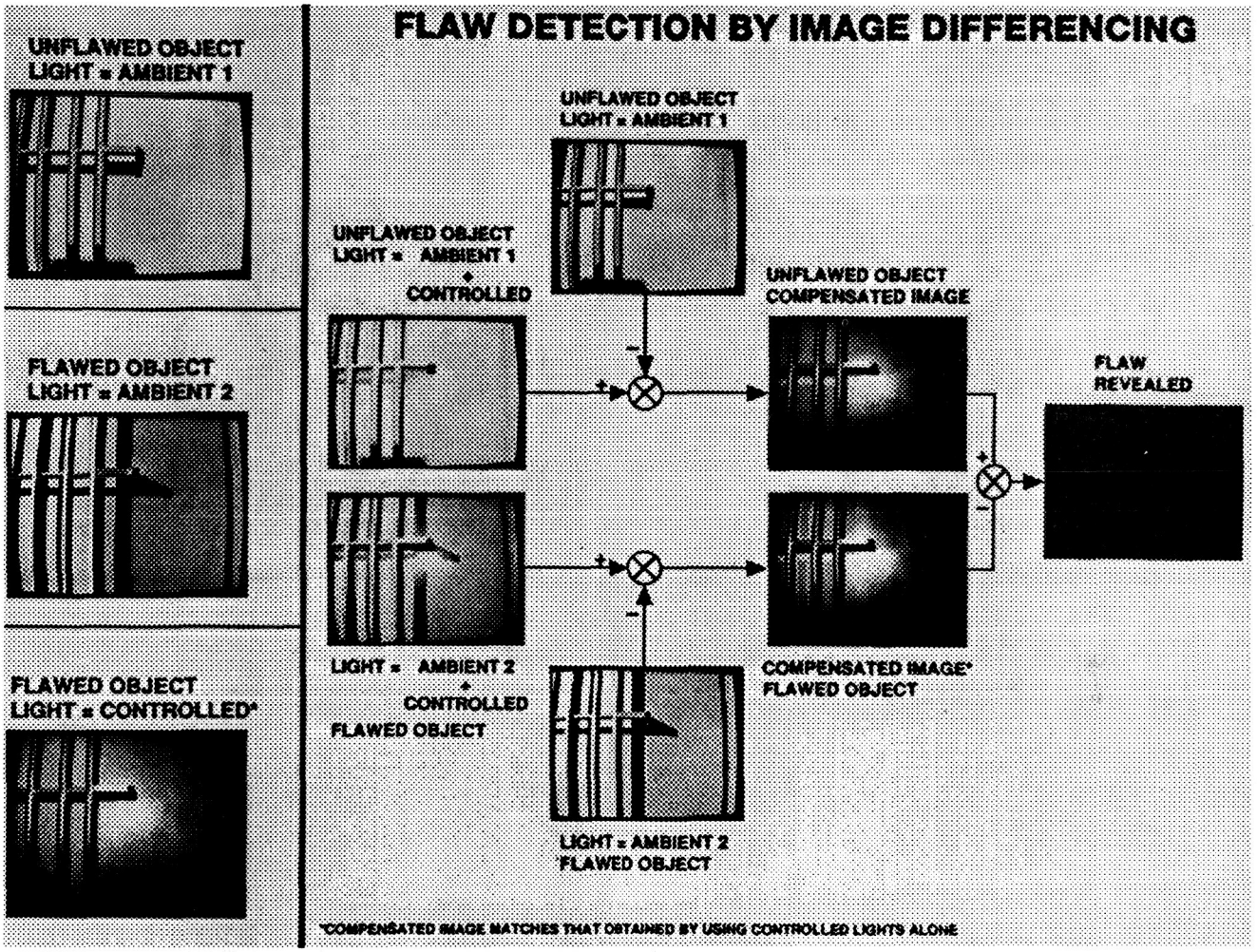


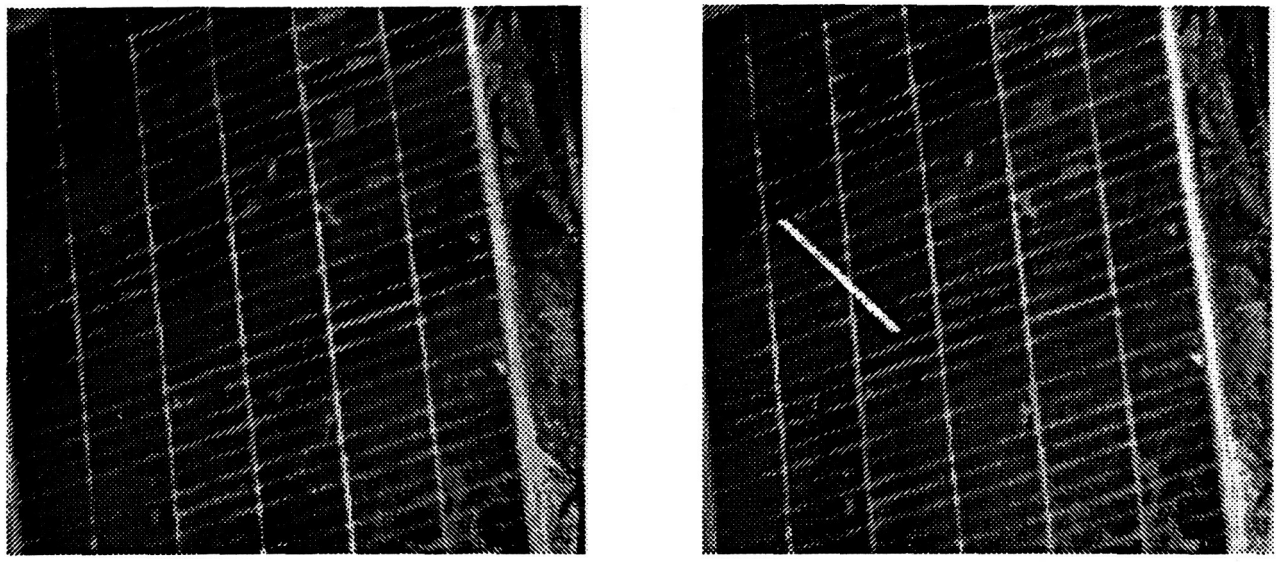
Figure 8. Flaw Detection by Image Differencing

5.2 Image Registration Accuracy

Image differencing technique suffers from the inherent accuracy limitations of moving camera platforms. In the laboratory environment, i.e., fixed targets and industrial arms with good repeatability, the inaccuracy translates to no more that one to two pixels when images are taken from relatively short distances of less than 2 feet. In the space environment, we expect to have larger repeatability problems due to arm flexibility, large thermal variations, and object location changes due to thermal and structural flexibility of the structure. It is therefore necessary to develop techniques to account for the misregistration before comparing before and after images.

We have devised several methods to estimate the registration error [13]. One approach is based on estimating the phase shift between the two images with spatial displacements in the frequency domain. It was found that this method can be used to reliably identify the lateral camera displacements (motion parallel to the image plane) in the presence of flaw and noise. This method is robust in the presence of noise but yields less accurate results when the image contains high frequency components. Figure 9 shows the before and after

images of the solar panel used to estimate the camera registration error. The white line in the second image represents a flaw. The dimensions are 200 X 300 pixels. The second image was displaced by -3 pixels in the horizontal and -2 in the vertical directions. The estimated shift was -2.25 and -1.9 in the horizontal and vertical directions, respectively. This results in sub-pixel registration correction which is the goal for the estimation process.



Original Image **Displaced Image with Flaw**

Figure 9. Solar Panel Images Used for Camera Registration Estimation

Two other approaches have been formulated and simulated for synthetic data. Application of these methods to real images are not complete. One approach is based on maximizing the correlation between the two images. The other is based on using the features on the images to estimate the camera position.

5.3 Efficient Approaches to Automated Inspection

Efficient automated inspection involves reliable flaw detection and robust flaw classification. The image differencing approach detailed above is clearly a useful first step in flaw detection. The method, however, does not address the important question of what *scale* the images should be processed at. The scale of an image is essentially the resolution associated with the image. It is well known [14] that relevant details of images exist only over a limited range of scale. Preliminary studies have been completed on estimating the image-scales relevant to flaw detection and classification. The essence of our approach is a multiresolution or "pyramidal" representation of the images. An initial high resolution image is successively convolved with a Gaussian kernel. The resulting images have decreasing spatial resolution and can be perceived as being stacked in a pyramid-like structure. The apex of this pyramid is a single pixel that represents the mean-pixel value of the original image which forms the base of the pyramid. The base level image, in addition to being noise prone, has a considerable amount of irrelevant information - that is the number of pixels associated with the image background is much greater that the number of pixels belonging to the image flaws. Efficient inspection, as we have learned from human inspection

strategies, involves discarding irrelevant information. Successive levels of the Gaussian pyramid from the base to the apex contain decreasing amounts of high frequency information. By taking the difference of two successive levels of a Gaussian pyramid, the "pass-band" content of the image can be estimated. This pass-band image is equivalent to performing an edge detection or a Laplacian convolution and reveals image structure information that is critical to flaw classification. A sequence of pass-band images, stacked to form a "Laplacian" pyramid reveals the range of scale that is most useful to flaw classification.

Three classes of images are being used to experiment with the above technique. One class is made up of large (≈2 m) field-of-view (FOV) pre-retrieval (in-orbit) LDEF surface images. These images were digitized from video sequences that were taped by the STS-32 shuttle crew during LDEF retrieval. The second set consists of medium (0.75 m) FOV images of laboratory ORU models. The third set consists of microscopic FOV (1 mm) images which were obtained from the LDEF post-retrieval image database at Johnson Space Center.

6. Experimental Results on Human Visual Inspection

An experiment was conducted to evaluate the usefulness of the system for the detection of damage caused by micrometeoroids [15]. A typical tank ORU was used for this task. The specific objective was to determine the time-to-completion and accuracy of inspection for micrometeoroid impact causing damage ranging from 1 to 10 pixels on the surface of the ORU. As part of the evaluation, we also compared the telerobotics-based inspection against direct human inspection. Since direct human visual inspection is unencumbered by the helmet that obscures EVA inspection, this provides a worst case test for telerobotics inspection. Table 3 shows the results of these experiments for the case of Teleoperated Human Visual Inspection discussed in Section 4.1. These preliminary tests show that remote inspection is approximately three times slower and 3 to 4% less accurate than direct inspection These results indicate that remote inspection can provide a safe and effective alternative to EVA inspection for a class of tasks.

Table 3. Experimental Results for Remote and Direct Micrometeoroid Inspection

Flaw Size	Remote Surface Inspection	Direct Inspection
Large Marks, 10 Pixel (2.7 Mm)	Time-To-Completion: 178 Sec Accuracy: 93%	Time-To-Completion: 57 Sec Accuracy: 97%
Small Marks, 1 Pixel (0.27 Mm)	Time-To-Completion: 308 Sec Accuracy: 91%	Time-To-Completion: 118 Sec Accuracy: 94%

7. Conclusions and Future Work

This paper has described the research and development effort for remote surface inspection at the Jet Propulsion Laboratory which started in 1991. The paper outlines the problem, the general approach and present initial results.

Future work will involve adding other manipulation and inspection sensors to the existing system for collision avoidance and for the detection of flaws which cannot be inspected by CCD cameras, such as gas leak, fine cracks, temperature variations, and so on.

The differencing approach will be further developed to use a scanning technique that will perform flaw detection continuously in real-time without stopping the arm to take images. Technique to Accommodate large misregistrations and to categorize flaw type are also being implemented.

8. ACKNOWLEDGMENTS

The research described in this document was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. We would like to acknowledge the contributions of M. Drews, D. Gennery, T. Lee, D. Lim, D. McAffee, R. Spencer, and R. Volpe to various aspects of the described work.

9. REFERENCES

[1] Fisher, F. and Price, C. R., "Space Station Freedom External Maintenance Task Team-Final Report," Volume I, NASA, Lyndon B. Johnson Space Center, Houston, TX, July 1990.

[2] See, T. et. al., "Meteoroid and Debris Impact Features Documented on the Long Duration Exposure Facility: A Preliminary Report," Space and Life Science Branch Publication # 84, JSC # 24608, August 1990.

[3] LDEF M&D-SIG Interim Technical Report, May 1992.

[4] LDEF 1992 conference abstracts

[5] Kinard, W., "Long Duration Exposure Facility Newsletter," published by LDEF Newsletter, P.O. Box 10518, Silver Spring, MD 20914.

[6] Mulholland, J. D., et al, "LDEF Interplanetary Dust Experiment: A High-Time Resolution Snapshot of the Near-Earth Particulate Environment," Proceedings, Workshop on Hypervelocity Impacts in Space, University of Kent at Canterbury, U. K., in press. 1991.

[7] Alan Watts (POD Associates, New Mexico), LDEF Conference 1992

[8] Seraji, H., "Configuration Control of Redundant Manipulators: Theory and Implementation," IEEE Trans. on Robotics and Automation, Vol. 5, No. 5, pp. 472-490, 1989.

[9] H. Seraji and R. Colbaugh, "Improved Configuration Control for Redundant Robots," Journal of Robotic Systems, 1990, 7(6), pp. 897-928.

[10] Krisch, J., "Inspection Robot," International Encyclopedia of Robotics, Edited by R. Dorf, John Wiley & Sons, 1988, Vol. 2, pp. 663-668.

[11] Isoda, K., "Advanced Robotic Inspection Applications," International Encyclopedia of Robotics, Edited by R. Dorf, John Wiley & Sons, 1988, Vol. 2, pp. 668-679.

[12] Sanz, J. and Petkovic, D., "Machine Vision Algorithms for Automated Inspection of Thin-Film Disk Heads," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 6, Nov. 1988, pp.830-848.

[13] VanNieuwstadt, M., "Algorithms for Automated Surface Inspection – Preliminary Report," JPL Internal Report, Oct. 22, 1991

[14] Koenderink, J, J., "The Structure of images," Biological Cybernetics, 50, 363-370 (1984)

[15] Gennery, D., "Results of Lighting Study," JPL Internal Memorandum # 347-2-91-067, Sept. 18, 1991.

SUPERVISORY AUTONOMOUS LOCAL-REMOTE CONTROL SYSTEM DESIGN: NEAR-TERM AND FAR-TERM APPLICATIONS

Wayne Zimmerman
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109

Paul Backes
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109

ABSTRACT

The JPL Supervisory Telerobotics Laboratory (STELER) has developed a unique local-remote robot control architecture which enables management of intermittent bus latencies and communication delays such as those expected for ground-remote operation of Space Station robotic systems via the TDRSS communication platform. At the local site, the operator updates the worksite world model using stereo video feedback and a model overlay/fitting algorithm which outputs the location and orientation of the object in free space. That information is relayed to the robot User Macro Interface (UMI) to enable programming of the robot control macros. The operator can then employ either manual teleoperation, shared control, or supervised autonomous control to manipulate the object under any degree of time-delay. The remote site performs the closed loop force/torque control, task monitoring, and reflex action. This paper describes the STELER local-remote robot control system, and further describes the near-term planned Space Station applications, along with potential far-term applications such as telescience, autonomous docking, and Lunar/Mars rovers.

INTRODUCTION

The original intent behind the design of the STELER local-remote robot control architecture was fourfold: 1) be responsive to recom- mendations coming out of NASA Space Station Extravehicular Activity (EVA) workload studies which strongly suggested ground-remote operations as a means of reducing EVA time and enhancing station utilization during the planned multi-year mantended phase (Refs 1, 2, and 3), 2) provide a modular design which would be more flight like (i.e., accommodate severe computational resource constraints at the remote robot site, manage either periodic remote site bus latencies (on the order of less than 1 sec) or manage major communication delays (greater than several seconds) between the ground and the remote site), 3) provide an easier means of transferring the technology to the flight centers, and 4) allow rapid tailoring or expansion of the system to other control applications. The current STELER local-remote design has evolved over a period of four (4) years.

Although the immediate application is Space Station telerobotics, the potential far-term applications include robot control/telescience between Earth and the Shuttle, Earth and the Space Station, and Earth and a rover on the Lunar or Mars surface. Other applications include autonomous docking, and terrestrial uses such as hazardous material handling, autonomous excavation, or undersea operations. Only the current planned work for space based applications will be discussed in this paper.

The Space Station robot control environment provides a good example of the intended application environment as a means of setting the stage for discussion of the local-remote control system design. In order to perform ground-remote operations the control system will have to manage communication delays on the order of 8 to 9 seconds (Ref 4). The majority of this delay is in the processing/packaging and relaying of the data from the ground, through TDRSS, then to the Space Station communication subsystem/operator workstation, and finally to the robot controller. To manage this time delay and provide robust control with limited on-orbit computational resources, the JPL STELER local-remote control architecture off-loads most of the high level work envelope, task assessment and planning functions to the local site. The local site uses low bandwidth stereo video images captured at the remote site from the robot/workcell cameras as the initial means of determining the actual position of objects in the environment (i.e., are objects where they are supposed to be, what is the viewing angle/ lighting environment like). The stereo video is displayed on a Silicon Graphics Inc. (SGI) workstation which also allows wire-frame, or solid, graphic models of objects in the workcell to be overlayed onto the video to verify the geometric model on the scene. The local site operator matches the graphic overlay with the video to determine the object's location/orientation and the world model is updated. This information is provided to the User Macro Interface (UMI) residing in the SGI where it is now used to parameterize the robot control macros to manipulate the object. The operator assembles the desired task control macros into a sequence and then sends them to the remote site Modular Telerobot Task Execution System (MOTES) where the commands are then executed and monitored at the robot site. MOTES is functionally equivalent to a spacecraft control system (e.g., Galileo). A command interpreter and dispatcher respectively manage incoming/ outgoing data. Sensor and Monitor modules process robot sensor data and check sensor thresholds/fault conditions. The Control module contains the trajectory generator and position based impedance control functions and generates the task space robot motion. A Fusion module merges motion commands from the various command sources (e.g., trajectory generator, hand controller commands, force sensor). A Device Driver module provides the low level communication to the physical components such as the robot arm and gripper. The next section provides a detailed discussion of the above system design elements and also describes the current/future laboratory hardware implementation.

LOCAL-REMOTE SYSTEM DESIGN

The above introduction briefly described the functions of the primary components in the local-remote robot control architecture. To lay the foundation for understanding planned applications, a more detailed system description is needed. This design description is provided in the following paragraphs.

A. Local Site System Design

The local site subsystem design has several components (Refs 5, 6) **Figure (1)** provides a schematic of the software modules which reside in the local site. **Figure (2)** shows the current laboratory operator workstation. Starting with the

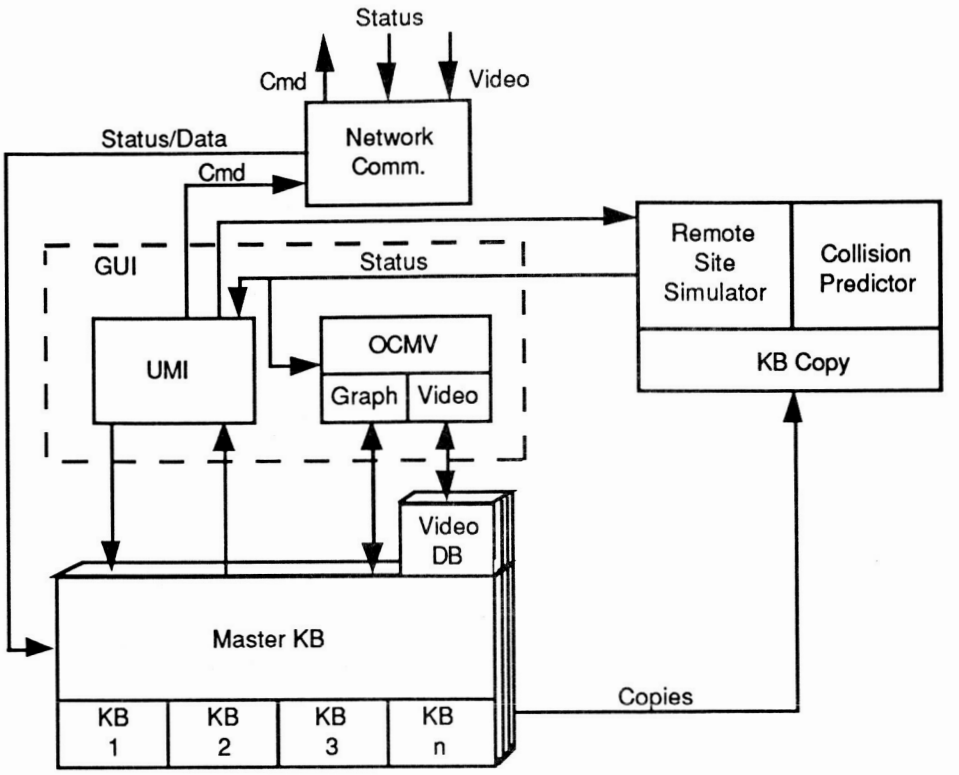


Figure (1) Local-Site Design

Figure (2) Current Laboratory Operator Workstation

operator interface, the operator accesses the various high level control menus and database through the keyboard, mouse, and spaceball input devices. The robot hand controller is the input device for controlling the manipulator(s) with tele-operation, and currently has a separate communication link to the MOTES subsystem at the remote site.

Moving to the next level, a **network comm** module with time delay buffer serves as the gateway for outgoing/incoming data between the local and remote sites, and for imposing variable time delay on return traffic. The hub of the local site revolves around the **user macro interface (UMI)** (manipulation) and **operator coached machine vision (OCMV)** (perception) modules, which reside within a high level **graphical user interface (GUI)** structure, and the **master knowledge base (MKB).** The local site presents to the operator a window-based GUI for commanding and sequence editing; and, a video/graphic interface that allows the operator to effectively perceive the task environment and to interactively modify the MKB. The GUI has four standard manipulation windows which are always visible-sequence, command, status/environment, and perception. The sequence window displays previously built sequences which are available. The command window displays the commands in the current sequence, and allows the operator to edit and create commands. Within the command window, a macro window displays the currently active control command/macro. The status/environment window displays the current system state information (e.g., which arm is active, is the system in "simulate" or "real" mode). The perception window provides the operator with options for updating video, graphics overlay, and changing the eye point of view. The GUI keeps track of, and displays, the current operation being performed and the most recent values of environment variables. **Figure (3)** shows a typical GUI display.

The UMI module is used to design, build, execute, store, and replay manipulation commands to be sent to the remote site for execution. The operator can interactively build/simulate complete task sequences using the menu interface provided by UMI. UMI provides parameter input checking, menus of robot control macros, sequence building, sequence simulation/execution, and status display.

OCMV contains both graphics and video elements. Using a menu interface, the operator can command video capture, thus allowing update of the local site video image of the workcell at the remote site. Once an updated set of images is returned and displayed, the operator can selectively relocate objects in the workcell using the graphic overlay capability. This is done by using either stored object models of known objects (called in from the knowledge base by clicking on the desired object in the video display with a cursor), or by performing an object build/edit on-line. The operator overlays and matches the object model with the video image using the spaceball. Once the overlay is properly mapped onto the video image, the machine vision object localization module determines the object's position/orientation using the internal camera models (e.g., viewing angle, focal point(s)) and position of the cameras relative to the workspace/object. The OCMV software also maintains known camera arm positions to allow different views of the workcell/

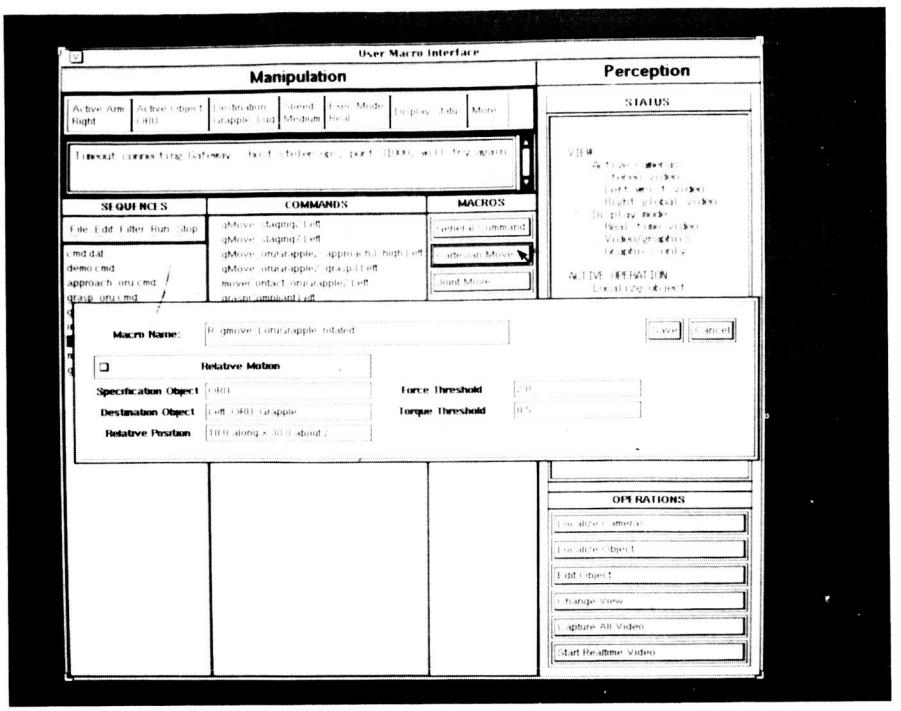


Figure (3) Typical GUI Display

object. **Figures (4) and (5)** display the graphics and video overlay capabilities respectively.

The last major component, the MKB module, is made up of nodes and links. Nodes represent real or fictitious objects; and, links represent relationships between the objects. A node contains object properties(e.g., polyhedral description, name, mass) and a link contains data about a relationship (e.g., object connectivity, coordinate transformations). The relationships between objects, either rigid or movable/breakable, are organized into parent-child links. The MKB also containsa master kinematic database, control/ dynamic parameters, simulator database (with collision prediction data), video database, and graphics database. The MKB also manages data storage/retrieval. The operator has the responsibility to maintain consistency within the MKB. This means that the MKB does not have an internal expert or rule based system which automatically checks and maintains consistency.

Figure (1) shows one more major component, the **remote site simulator and collision predictor (RSSCP).** In the near term, the remote site simulator located in MOTES will be called from the local site to test a task sequence before actual execution. The status will be returned to the local site and viewed with the graphics simulation. Eventually, a copy of the remote site simulator will reside at the local site and will simulate not only the system state transitions, but also the kinematics and simple force contact models (e.g., magnitude but not vectors). The

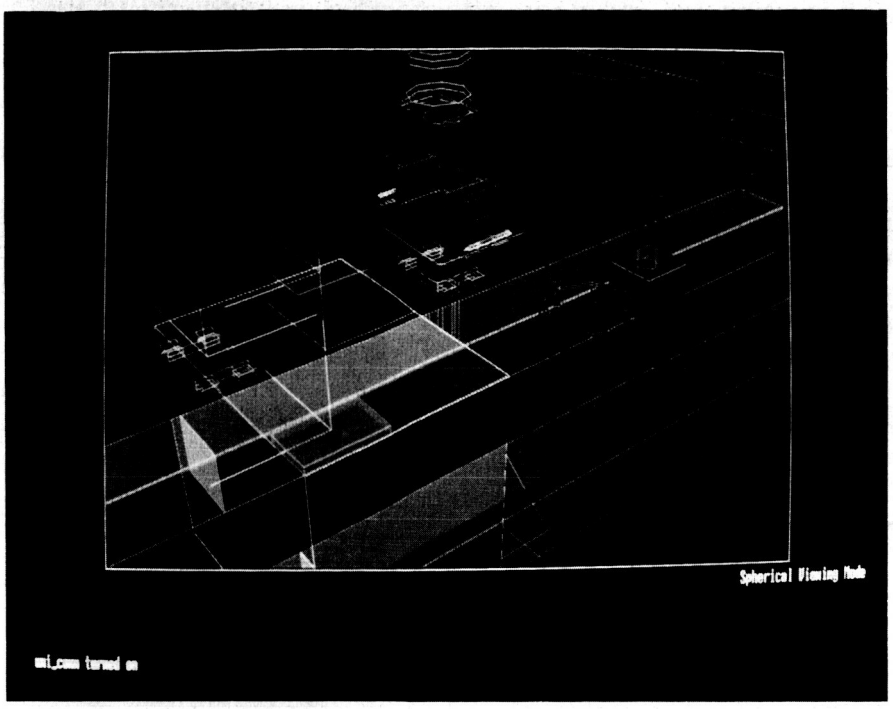


Figure (4) Perception Graphics Overlay

Figure (5) Perception Graphics Overlay on Video

collision predictor takes trajectory generator via points, arm kinematics, and global geometry and checks for collisions with objects in proximity to the joints as the trajectory is executed in the simulator. Collisions are highlighted and an eventual safe trajectory is iteratively generated.

B. Remote Site Modular Telerobot Task Execution System (MOTES) Design

The MOTES robot control design has been discussed in detail in other literature (Ref 7), and will therefore, only be summarized here. As stated above in the Introduction, MOTES has several modules. Each module interfaces to the rest of the system through shared memory with specified input and output parameters/functionality. **Figure (6)** provides a functional diagram of MOTES. The various modules operate asynchronously with the **executive** module handling communication with the local site, placing new commands into the task/

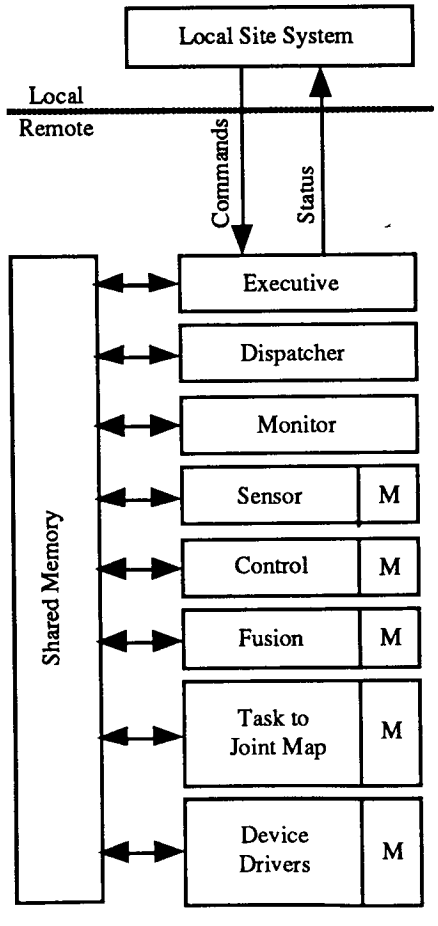


Figure (6) MOTES Remote Site Functional Diagram

reflex command queues, and returning status data to the local site. The **dispatcher** module checks/controls the transition between execution states by monitoring the status of the various modules and specifying the appropriate commands/parameters to the various modules via shared memory. The dispatcher controls the transition between commands by changing the active command block parameters as specified by a monitor event (e.g., completion of a command sequence, setting of an error flag causing a halt in command execution). If an event terminates under an acceptable condition (e.g., trajectory successfully executed), then the next command in the queue is issued. If an unacceptable termination condition results, then the dispatcher finds which reflex command queue to use based on the matching flag set in the reflex table associated with that event. The dispatcher executes the set of reflex commands, clears the task command queue, and sends a report back to the exec stating which reflex action was initiated.The **monitor** module(s) check the behavior of the system and set the appropriate success/error flags when a monitor event has occurred. Examples of monitor events were provided in the preceding paragraph. Monitoring occurs at two different levels: 1) the system level (e.g., multi-sensor based monitoring, command completion), and 2) the sensor/reflex level (e.g., force threshold exceeded). In designing the system, previous experience dictated that it is important to monitor and control low level events where the information is generated and the context is known. This design also improves response time for critical events.

The **sensor** module(s) generates either real (e.g., force/torque sensors, encoders, proximity sensors), or virtual (e.g., bounds on distances to joint limits, repelling forces when within a given distance from collision points), sensor data which is placed in shared memory for use by the other MOTES modules.

The **control** module(s) generates manipulator setpoints, and performs control based on both the local site generated command sequence and sensor data. Each control module has a uniquely specified Cartesian control frame for control of the manipulator. The complete motion command specified by each command module is transformed to a common task space (consisting of the common Cartesian frame and arm angle) before being placed in shared memory. The position trajectory generator is the control module which computes the desired task space path including Cartesian frame and arm angle.

The **fusion** module combines the commands from the various control modules as specified in the task parameters sent by UMI. For sensor based control, the trajectory generator employs a position based impedance model in each degree of freedom of the task space. The fusion module also allows position based inputs from the teleoperator to be fused with autonomous sensor based control inputs (provided by the monitor/control modules). The subsequent **task-to-joint map** module maps the task space command of the fusion module to the actuator space of the physical device being controlled.

Last, the **device driver** module is responsible for communicating commands, and status, with the actual system hardware as well as performing hardware specific

computations. The primary device drivers are for the manipulator (Robotics Research Arm), force/ torque sensor (Lord), and servoed gripper (Telerobotics Research Inc. (TRI)).

C. Current/Future System Implementation

To set the final stage for discussing applications, it is important to understand the current hardware/software implementation envir-onments. The local site system currently runs predominantly on an SGI 310/VGX workstation, using the IRIX 4.0.4 operating system. The SGI workstation was selected because the target Space Station Mission Operations Planning environment at Johnson Space Center (JSC) for telerobotics is the SGI. A Sun 4/260 workstation at the local site is used as to develop and test the remote site MOTES real-time Ada code. As stated earlier, in the near term the remote site simulator running in MOTES will be used to simulate task sequences. However, it is planned to eventually test MOTES and simulate task sequences using a carefully descoped version of MOTES running on the Sun 4 (i.e., the sequence error checking, trajectory generator, and position success/error flags are the only predominant features needed for local site simulation). In the future, it is desirable to port all local site software to the SGI. To round out the local site hardware, a 6DOF JPL/Salisbury Model C hand-controller and motor controller and associated VME chassis allow teleoperation of the 7DOF arm. An indexing trigger on the hand controller provides the additional means for controlling the redundancy, or seventh degree of freedom. MOTES is written in Ada and runs in a VME environment on Heurikon 68020 processors. A Verdix/ VADSworks Ada cross compiler is used to produce the target code. The target code runs on top of the VADS-works operating system in the 68020 environment. The Verdix/VADSworks Ada compiler and 68020 environments were selected primarily because they provided the desired performance for the least investment. Additionally, since the Space Station software environment will be Ada, it was decided that by moving to real-time Ada the system would be more in synch with the planned flight software for the station remote manipulator and the Canadian Special Purpose Dexterous Manipulator (SPDM). Communication between the 7DOF Robotics Research arm (model K1207) controller and the MOTES VME chassis is through a pair of Bit-3 memory interface cards. The direct servo control of the arm joints is done through the factory supplied controller.

Since the 68020 CPU's used in this initial version of MOTES are relatively slow (the 68020 V2FA runs at approximately 1.7 MIPS), a multiple CPU environment was required. This decision to go to a multi-processor environment was also considered viable because the SPDM design is currently a multi-processor configuration. Therefore, it was considered useful to explore and resolve potential scheduling/timing problems associated with real-time control in a multi-processor asynchronous environment. Figure (7) displays the current VME multi-processor environment running in the lab. Although the system is operational, based on current test results, it appears that it is desirable to reduce the number of multi-process-or boards from the present five to one, or two. To achieve this, the new MOTES design in considering an 80386 configuration, running the commercially available Lynx operating system. This configuration will condense MOTES, increase performance, and also provide a compact flight qualifiable controller for

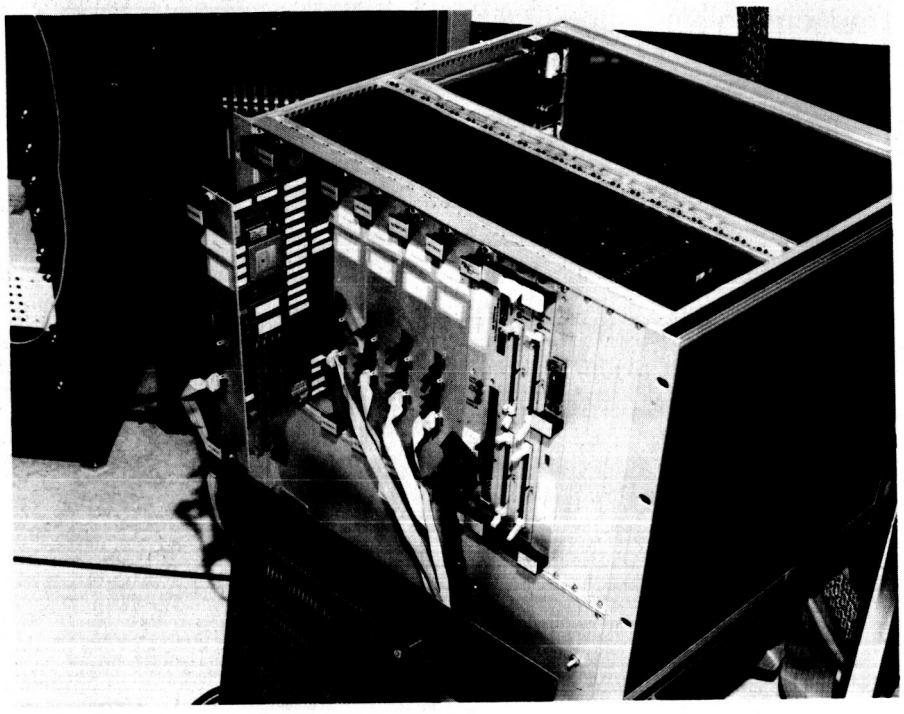


Figure (7) STELER MOTES Remote Site Controller Hardware

systems. As will be seen in the following discussion, the planned evolutions of both the local site (full SGI implementation) and remote site (80386 flight qualifiable configuration) will be important to achieving the planned applications.

PLANNED APPLICATIONS

A. Near Term

The current JPL STELER local-remote control system has several immediate applications relative to the Space Station flight program. First, before the station is ever launched it is important to start the process of assembling and testing robot control sequences for station/payload servicing now. These sequences must be tested on the ground under high fidelity task simulation conditions in a laboratory (e.g., lighting/lighting angle, object/shadow occlusions). Further, to the degree possible, similar robotic devices (SSRMS/SPDM) and control environments should be used to duplicate operator/system control response. Although the degree to which the U.S. NASA community is planning to perform task verification is still uncertain (i.e., partial verification using graphics simulation vs. full task verification in a laboratory with real robotic/task hardware), it is still important to have the technology ready soon to enable start of task sequence assembly/testing. The JPL system is currently on schedule for delivery into the JSC Space Station ground test/evaluation laboratory by the end of FY93. As a spin-off from the above ground

application, the system (which allows either full manual teleoperation of the robot(s), shared control, or supervised autonomy) also provides an essential training medium for astronauts from the standpoint of both hands-on control, and assessing human-machine tradeoffs in the performance of various tasks under varying environmental conditions. For example, the astronaut operators may determine that for some manipulation tasks in complex/occluded environments, it is more efficient/safer to have the operator perform the gross motion control while allowing the autonomous system to set the arm pose and perform the proximity/fine alignment control.

The most powerful application of the STELER local-remote robot control system is for ground control of on-orbit station robots. The recent mid-year review of the JPL project status, by NASA headquarters, revealed that the downscoping of the station (and subsequent reduction in astronaut crew) has still not resolved the substantial projected on-orbit servicing/payload workload (Ref 8). As a result, the most logical way to reduce on-orbit workload and increase station utilization (i.e., in the early man-tended phase of operation, the Space Station will not be permanently manned and will therefore see an estimated utilization on the order of 13%) is to perform simple setup and inspection tasks from the ground. To this end, the local-remote control system has been tested in actual hardware/software segregation scenarios such as 1) the control of an industrial robot inspecting a Shuttle payload rocket booster at Kennedy Space Center, from JPL's STELER lab located 3000 miles away using an early local-remote architecture functionally equivalent to the current design (time delay of 2-3 seconds) (Ref 9), 2) control of the JPL STELER robots from a temporary workstation located in a trailor approximately 100 yards away (imposed time delay ranging from 0-9 seconds) (Ref 10), and 3) control of a PUMA robot simulation located at Texas A&M University, from JPL, via the Internet/TelRIP communication network (time delay of roughly 1-3 seconds) (Ref 11). It is also planned to control the JSC Robotics Research arm from JPL as well, and use the Internet/TelRIP interface to both send and debug software used in the JSC telerobot test/integration laboratory. Again, by streamlining the hardware/ software architecture, the transfer of this technology will be greatly simplified.

The last major near-term application of the JPL local-remote architecture is somewhat related to the earlier ground-based system testing/training, but applies to on-orbit robot operations. Currently, it is planned to control the station robotic systems primarily in teleoperation mode. However, due to limited camera views, potential lighting/occlusion problems, and, most importantly, limited on-board computational resources, teleoperators will need to depend greatly on ground task simulation/verification before proceeding. The STELER local site on the ground will provide a means for accurately updating the ground workcell model even in the presence of limited viewing/occlusions. The local-remote site combined will provide a high-fidelity simulation of the task, and assembly of the ultimate control macros which get telemetered up to the station teleoperator workstation for the astronaut to use during task execution as a control template.

B. Far Term

As stated in the "implementation" subsection, it is planned to move to more "flight like" versions of the local-remote site systems by implementing the local site completely on a SGI workstation, and the remote site on an 80386 computational environment. Work just being initiated for FY93 in STELER calls for an expansion into the new area of telescience. New work will start on the design and construction of a "mini-device" (called the Lab Tending Telerobot (LATT)) which can be placed as a module on the front of a science rack, or inside, and which can be programmed/controlled from the ground by the principal investigator (PI). The device will allow the PI to physically interact with the science experiment(s) on-board the Shuttle (intermediate term target), and eventually the Space Station or free flying platforms (far term target). The more compact hardware/software implementation will allow LATT ground and on-orbit elements to be respectively, 1) easily ported to the PI's base facility, and 2) more easily space qualified since the 80386 Intel processor has already been space qualified.

Another more intermediate-to-far term application is for autonomous docking. As a test of the STELER control system robustness, an autonomous docking task has been structured. The female fixture is mounted on a full scale satellite mockup which hangs freely from a cable and has five degrees of freedom. The male fixture is mountable on the 7DOF arm, or on a jig fixture. The control scenario calls for the local site to update the position of the fixture on the satellite, followed by either the manipulator acting as the chase vehicle and autonomously docking with the satellite; or, the arm grasping a grapple fixture attached to the satellite and moving the satellite to dock with the jigged male half. Work planned for FY93 calls for the STELER team to begin building the appropriate graphics, simulation, and control macros for an autonomous docking workstation running on the SGI, with eventual transfer to the Marshall Space Flight Center (MSFC) docking simulation facility.

The last long term application of the local-remote control system is for Lunar/Mars vehicles. For potential early precursor missions, either autonomous mini or micro-rovers may be used to map potential landing sites and perform science experiments (e.g., soil sampling, coring, experiment module placement). With time delays on the order of many seconds (Lunar) to many minutes (Mars) it is imperative that a local-remote control architecture like STELER's be used to insure robust/safe control of the robotic vehicles and manipulation functions. The current move towards a more efficient, compact remote site controller design is the next step towards learning how to provide robust remote site control/intelligence within tightly constrained volume/mass requirements.

CONCLUSIONS

A description of the current JPL STELER local-remote telerobot control system was provided along with present and future hardware/software architecture implementations. Additionally, a discussion of both near and far term applications was provided and mapped onto the planned hardware/software evolutions. Most

importantly, this paper provided a roadmap and structure on how the STELER team plans to carefully evolve, test, and implement the local-remote control capability towards an expanding application domain.

ACKNOWLEDGMENTS

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

1. Zimmerman, W., Swietek, G., "Factors Pointing to Evolution, A&R," NASA, 1989 (Informal document).
2. Weeks, D., Zimmerman, W., Swietek, G., "Space Station Freedom Automation and Robotics: An Assessment of the Potential for Increased Productivity," NASA/MITRE, December 1989.
3. Fisher, W., Price, C., "Space Station Freedom External Maintenance Task Team Final Report," Vols. 1 and 2, NASA, Johnson Space Flight Center, July 1990.
4. Aster, R, Pitahaya, J., Deshpande, G., "Analysis of End-to-End Information System Latency for Space Station Freedom," JPL, Doc. no. D-8650, May 1991 (Informal Document).
5. Beahan, J., "Local Site Software Design," JPL, June, 1992 (Informal Document).
6. Bon, B., Beahan, J., "A Graphics Based Operator Control Station for Local-Remote Telerobotics," SPIE, April 1992.
7. Backes, P., Long, M., Steele, R., "Designing Minimal Space Telerobotics Systems for Maximum Performance," AIAA, February 1992.
8. Zimmerman, W., "JPL Space Station Engineering Prototype Development (EPD) Program: Telerobotics Mid-Year Review, JPL, NASA HQ Level I/II, June 1992 (Presentation).
9. Bon, B., et. al., "Telerobotic Control from 3000 Miles," JPL/NASA Kennedy Space Center, November 1989 (Laboratory Demonstration).
10. Zimmerman, W., Backes, P., Bon, B., Long, M., Steele, R., "JPL FY91 End-of-Year Telerobot Control Review," November 1991 (Laboratory Demonstration).
11. Tso, K., "Remote Control of Texas A&M PUMA-Simulation," March, 1992 (Laboratory Demonstration).

PERFORMANCE ANALYSIS OF A ROBOTIC SYSTEM FOR SPACE-BASED ASSEMBLY

Prof. Alan Desrochers and **Hauke Jungnitz**
Rensselaer Polytechnic Institute
CII 8015
Troy, NY 12180-3590

A dual arm system for space-based assembly has been under development and experimentation at the Center of Intelligent Robotic Systems for Space Exploration. This paper will describe the components of this system and will then focus on the specific problem of predicting response time of the integrated system. This hierarchical system consists of a task representation and planning level, a coordination level, and an execution level to carry out the desired assembly task. A common modeling methodology based on Petri nets has been used to model the various subtasks throughout the levels of the hierarchy. The Petri net model can then be analyzed for liveness and boundedness that, in turn, are used to detect the existence of deadlock or conflict in the system. The model can also be used to calculate the time it will take to complete an assembly task once an operator issues a command. These modeling results are compared with the actual experimental results obtained from carrying out the assembly task.

Session R2: FULLY ROBOTIC SYSTEMS

Session Chair: Capt. Ron Julian

N93-32102

AUTOMATED ASSEMBLY OF LARGE SPACE STRUCTURES USING AN EXPERT SYSTEM EXECUTIVE

Cheryl L. Allen; NASA Langley Research Center; MS 152D; Hampton Va 23665-5225

ABSTRACT

NASA Langley Research Center has developed a unique testbed for investigating the practical problems associated with the assembly of large space structures using robotic manipulators. The testbed is an interdisciplinary effort which considers the full spectrum of assembly problems from the design of mechanisms to the development of software. This paper will describe the automated structures assembly testbed and its operation, detail the expert system executive and its development, and discuss the planned system evolution. Emphasis will be placed on the expert system development of the program executive.

The executive program must be capable of directing and reliably performing complex assembly tasks with the flexibility to recover from realistic system errors. By employing an expert system, information pertaining to the operation of the system was encapsulated concisely within a knowledge base. This lead to a substantial reduction in code, increased flexibility, eased software upgrades, and realized a savings in software maintenance costs.

INTRODUCTION

Projected crewed missions to the moon and Mars represent a departure from previous space endeavors in that the large vehicles involved will have to be assembled and checked out on orbit. The construction of these vehicles will require extensive in-space operations calling for enhanced capabilities in the areas of assembly and servicing. In order to perform these functions with the limited crew resources available, a much higher level of automation must be realized than is currently obtainable. NASA Langley Research Center has developed a unique testbed to investigate the practical problems associated with the automated assembly of large space structures using robotic manipulators. The research program is an interdisciplinary effort which considers the full spectrum of assembly problems from the design of mechanisms compatible with automated operations to the definition of software structures and algorithms required for their support.

The LARC research program adheres to several principles and ground rules: (1) all system development, testing, and demonstration is performed using realistic test hardware, which is felt to be the only way to identify all the problems associated with automated assembly; (2) system design and automation are considered integrated and complimentary technologies with solutions developed cooperatively; and (3) the program is targeted towards a fully automated system that utilizes either an astronaut or earth based operator as a monitor who is called upon only when the robotic system encounters a problem requiring intervention or assistance. The third principle describes a mode of operation known as supervised autonomy which holds the most promise for the accomplishment of large construction tasks with the limited crew resources available on orbit.

The purpose of this paper is to briefly describe the automated structures assembly testbed and its operation, to detail the expert system executive and its development, and to discuss the system expansion currently underway. The emphasis of the paper is on the expert system implementation of the program executive, however the system components are described and a narrative of the assembly process is given to serve as a basis for the description of the software and its functions.

FACILITY DESCRIPTION

The Automated Structures Assembly Laboratory (ASAL) is shown in figure 1. Figure 1a shows a schematic of the assembly system with the major components labeled, and figure 1b is an actual photo of the facility in operation. The assembly system consists of a robot arm, a motion base system, two specialized end effectors, components for a truss assembly, and storage canisters for those components. The ASAL utilizes commercially available equipment to minimize cost and ease modification as research needs dictate. The hardware system is a ground-based research tool designed to permit evaluation of assembly techniques, strut and end effector components, computer software architecture and algorithms, and operator interface requirements.

The structure selected for assembly is a planar tetrahedral truss which supports hexagonal reflector-type panels (see figure 2). The completed structure consists of 102 two-meter-long strut members and 12 panels measuring approximately 2.3 meters across the vertices. The structure was designed to be a laboratory prototype representative of the type of structures which support the functional surfaces of a number of planned or proposed missions, such as antennas and aerobrakes.

A brief description of the major components will follow; however, the details of the facility hardware, performance characteristics, and assembly procedures can be found in references 1-3.

Robot Arm

The robot arm is an electronically driven six-degree-of-freedom industrial manipulator selected for its reach envelope, payload capacity, positioning repeatability, and reliability. The robot arm computer is based on a 68000 microprocessor and all robot motions are programmed in a modified BASIC programming language. No modifications have been made to the manipulator other than those available from the manufacturer.

Motion Base System

The motion base system includes a linear translational x-y Cartesian carriage and a rotating turntable. The robot arm is mounted on the carriage, and the truss is assembled on the rotating turntable located at one end of the x carriage. Motion base drive motors on all three axes are commanded by a 80286 micro-processor based indexer.

End Effectors

The end effectors, as shown in figure 3, are specialized tools mounted on the robot arm which perform the strut and panel installation and removal operations. Figure 3a shows the strut end effector and figure 3b shows the panel end

effector. All end effector operations are controlled by an onboard microprocessor mounted near the robot wrist. Typical microprocessor operations are detailed in reference 4. All end effector mechanisms are equipped with simple sensors such as microswitches and linear potentiometers to monitor operations and notify the operator if a problem occurs. The processor is programmed in ANSI compatible C and includes sufficient I/O to monitor the sensors associated with the end effector mechanism operations. A commercial force/torque load cell is mounted between the end effector and the robot arm to provide compliant move capability during both strut pick-up and installation operations.

Truss/Panel Elements

The truss joints and nodes designed for this assembly are shown in figure 4. The joint is composed of two parts, a connector section which is bonded to the graphite-epoxy tube to form a strut and the receptacle section which is mechanically attached to the node. The truss members are connected by specially designed connector joints located near the nodes. The strut end effector grasps and holds the joint receptacle to provide stability during strut installation and removal (ref. 5). The strut end effector uses pneumatically actuated receptacle fingers to grasp passive guidance v-grooves on the node receptacles. After the end effector inserts the strut into the receptacle, locking nuts are turned by a small electric gear-head motor, securing the strut into place. Assembly begins by connecting struts to three nodes that are premounted on the motion base turntable.

As the truss assembly progresses, the panels are placed on the nodes at the top of the truss using the panel end effector (ref. 6). The panel is an aluminium hexagonal frame with a reflective mylar covering. Once in position the panels are locked into place using end effector actuator pins.

Storage Canisters

The truss struts are stored in nine trays which are stacked in the working canister directly behind the robot arm. Each tray is fitted with handles which allow the strut end effector to pick up empty trays from the working canister and transfer them to the storage canister located to one side of the robot arm.

The panels are stored vertically in a large canister at one end of the y carriage. The same pins that are used to attach the panels to the truss structure are also used to latch the panels in the canister.

ASSEMBLY PROCEDURE

The assembly process begins when the strut end effector acquires the first strut from the top tray in the working canister. Once acquired, the strut is carried above the working canister and the motion bases are positioned so that the robot arm can reach the required installation position. The robot arm then moves through a sequence of predetermined points, arriving at an approach point located approximately 12 inches from the intended installation point in the structure. At the approach point, control is turned over to a machine vision system.

The machine vision system uses two small video cameras located on the end effector to view targets made of reflective material mounted on the node receptacles as shown in figure 5. A special five dot domino pattern is used as the target. The video image of the target is processed to discriminate the target from the background and the centroids of the dots are determined. The position of the centroids are defined with respect to the camera using a pose estimation routine. The pose information is used to direct robot arm moves toward the target location for strut installation. Details of the vision system can be found in reference 7. Once the arm reaches the installation point, the vision system relinquishes control and the end effector grapples the strut receptacles in the structure, repositions the robot arm to reduce forces and torques at the end effector that are caused be minor positioning errors, and inserts and locks the strut joint. The robot arm then returns to the working canister for another strut.

Once a specified number of struts have been installed, panels can be secured to the top of the structure. This involves stowing the strut end effector by latching it to the tray in the top of the storage canister and picking up the panel end effector stored at one end of the panel canister. This end effector change is accomplished by a commercially available pneumatic quick-change mechanism. Panels are retrieved using y-carriage motion base moves, and installed at predetermined points atop the structure. Machine vision is not used for the placement of panels in the structure.

Combinations of strut and panel installation sequences are currently followed until a platform with 102 struts and 12 panels is completed.

SYSTEM CONTROL AND COMMUNICATIONS

The ASAL facility is managed by several digital computers serially connected through RS232 communication lines as depicted in figure 6. The system executive and operator interface functions are performed on a micro-VAX workstation. The robot motions, carriage movements, and end effector operations are executed on individual processors, as are the computations required by the vision system.

Software Design

The design layout for the assembly system software is illustrated in figure 7 and detailed in reference 8. The software is arranged into four hierarchical levels of commands (Administrative, Assembly, Device and Component) each of which decompose into a sequence of commands for the next lower level. The highest or administrative level performs the preliminary setup of the system. The operator can examine and modify data and system options. Command and assembly sequence files can be selected, created and modified. It is this level that is intended to interface with a goal-directed task sequence planner. Currently the assembly sequence is manually determined and maintained in a file. Each entry in the assembly sequence file represents an appropriate assembly level command which specifies the operations to be performed on a given element (ie. strut or panel). The standard operating mode is centered at the assembly level and reflects the automated aspect of the system. At this level the software manages all the devices, data verification, and error recovery. The assembly level commands decompose into a series of commands for each of the three devices; the motion bases, the robot arm, and the end effector.

Although the assembly software system is intended to operate in a fully automated mode, it is imperative that the operator be provided with sufficient internal information and have command access and authority at all levels to deal effectively with assembly errors. The operator has complete control of error recovery and final decision on error resolution. The operator may decide that an error is not severe and command the system to proceed anyway. Also, if none of the recovery options presented are successful, the

operator may instruct the system to abort the failed operation and automatically roll the assembly process back to a known, successful condition. During assembly operations, the operator has the capability to pause the assembly process at any point and observe some detail using a video display before either continuing or reversing the sequence. This intervention capability imposes a significant burden on the system software.

The system executive directs and monitors assembly operations across the various processors, and reports current status information to the operator. The executive maintains the conditions and constraints of the assembly operations, including details of the geometry of both the structure and the storage canisters. During an assembly, the executive makes decisions about what end effector to use and the procedures required for it's use. Finally, the executive keeps track of possible problems and recovery techniques for all assembly scenarios. In order to do this effectively, the executive has full access to the current status of the assembly operation and the system hardware including complete, detailed descriptions of the state of the assembled structure, the motion base, the robot arm, and the end effector hardware. This information is continuously updated based upon verification by sensors.

Initially, the assembly executive software was written in FORTRAN. The procedural language was already familiar to developers in ASAL and therefore could be used to verify and refine assembly system operations in a relatively short period of time. The initial task was to construct a simplified structure of 102 struts, using a single, premounted end effector who's functions were commanded via the robot arm computer. The robot arm moved to predefined installation positions without the use of machine vision. As the scope of the research project grew (with the addition of panels, a second end effector, and distributed processors) the complexity of the knowledge to be managed by the assembly software increased. Because traditional programming languages proved to be cumbersome in keeping pace with system upgrades, the decision was made to rewrite portions of the software using an expert system. The first level of code targeted for this transition was the decision-intensive assembly executive. The following sections describe the application of expert system techniques to the assembly executive, giving examples of their use.

Expert System Assembly Executive

An expert system is a computer program that uses knowledge and reasoning techniques to solve problems that normally require the services of a human expert. Like conventional programs, expert systems usually perform well defined tasks; however, unlike conventional programs, expert systems also explain their actions, justify their conclusions, and provide details of the knowledge they contain. Expert systems are ideal for capturing and utilizing the "rules of thumb"-type logic that evolves from the experience gained in ASAL.

The assembly executive is responsible for making decisions about the actions to take (and the order to take them) during the construction of a given structure. To make informed decisions, the executive has to have access to all current system information, and the knowledge to evaluate that information in light of the desired task. It is this decision-making component of the assembly executive that was best suited to implementation using expert system techniques.

Methodology

A subset of the general area of expert systems concentrates on explicitly representing an expert's knowledge about a class of problems and then providing a separate reasoning mechanism (called an inference engine) that operates on this knowledge to produce a solution. These kinds of systems are known as knowledge-based expert systems. The knowledge base is a file containing the facts which make up the human expert's knowledge about a specific domain. An inference engine is a program that applies reasoning techniques to the facts, as defined by the knowledge base, to draw conclusions. Inference engines vary according to the representation of the knowledge and the strategy for applying the knowledge.

There are a variety of expert system development tools available to assist programmers in building powerful systems capable of solving a wide range of problems. The commercially available Knowledge Engineering System (KES (ref. 9 and 10)) was selected for use in ASAL. The KES tool provides the inference engine, knowledge representation schemes, and facilities for creating an operator interface. KES provides an embedding technique for integrating expert systems with existing software by allowing procedural language code to send, receive, and modify data from a knowledge base through the use of special data types and run-time functions.

The KES inference engine uses rules to represent knowledge. This knowledge representation scheme is particularly well-suited to applications such as automated assembly where the facts can be organized in the form of branching logic or if-then constructs. KES uses deductive reasoning as the technique for problem solving, where certain outcomes follow directly from certain inputs.

The pursuit of a solution (or goal) drives the reasoning methodology used by KES. This goal-driven inferencing technique is known as backward chaining. Implicit subgoals are set up to determine values for attributes that appear in the antecedent of a rule that infers a value for some other attribute, and so on, until a value for the goal attribute has been determined. In addition to goal-driven inferencing, KES also performs event-driven inferencing through the use of demons. Event-driven (forward chaining) inferencing takes place when the expert system responds to the occurrence of an event rather than the pursuit of a goal.

The following section will describe how the fore-mentioned methodologies have been applied to the automated assembly system software in ASAL using KES.

Implementation

As mentioned previously, the executive portion of the assembly system software was the first to be implemented as a expert system. The executive is responsible for managing all the devices (the motion bases, the robot arm, the end effectors, and the vision system), data verification, and error recovery. Figure 8 illustrates where the knowledge base fits into the overall software system architecture. By embedding the knowledge base in the automated assembly system , the executive has access to expert system methodologies for decision-making while leaving the already familiar operator interface and existing database management schemes intact.

The operator gains access to the executive through a menu-driven interface. By implementing a menu-driven interface, the operator is only presented with the commands he needs at any given time. As shown in figure 8, a layer of procedural code (FORTRAN and C routines) surrounds the knowledge base and handles the menuing functions and information exchange between the knowledge base and the

hardware. Database information is also transferred through this surrounding code. The knowledge base contains the data constructs (attributes and classes), rules, and demons necessary to make informed decisions about assembly actions.

The expert system uses the knowledge base as the primary source for determining the command sent to a particular device at any given time. Commands are sent to the individual processors associated with the specific hardware device for interpretation and execution. When up-to-date information about a piece of hardware is needed, sensors are polled through the device interfaces and the information is passed back to the knowledge base. After a device-specific processor has completed processing a command, a return status is forwarded to the knowledge base so the next action can be sent. In the case of a successful return, the database is updated and the next command in the sequence of assembly actions is determined. In the case of an error, instructions to return to the last known successful state may be issued. Information about all system functions is constantly updated and reported to the operator via status windows.

The structuring and content of the knowledge base lies at the heart of the expert system, and therefore warrants further consideration. The next sections will detail the more important components of the knowledge base, and present examples of their application.

Classes-

KES uses a structure called a class to describe a group of objects having the same set of characteristics. Each object is referred to as a member of the class, and each characteristic is maintained in a class construct known as an attribute. Two classes are defined in the current automated assembly knowledge base; one for struts and one for panels.

There are 102 unique members in the strut class; one for each strut in the truss assembly. An example of the attribute declarations for the strut members is shown in figure 9. The values associated with these attributes are stored in a database and are associated with some physical aspect of the strut and the way it is stored in the canister or installed in the structure. As indicated in the figure, there are 13 attributes identified for struts: three associated with naming conventions (OBSERVER NAME, ALTERNATE NAME, and ROBOT NAME); two identifying the canister storage location (TRAY, SLOT); five containing information about the physical characteristics of the strut (NODE END) and any special conditions for installation (CAP END, FLIP, CAN_FLIP, and NODE DIRECT); one to track the current location of the strut (WHERE); and two that define carriage positions of the robot during installation (MB_INDEX1 and MB_INDEX2). Additional information regarding these attributes can be found in reference 8.

A class has also been defined for panels and contains information pertaining to the installed location for the panel and the whether or not the panel is installed.

Rules-

Rules are the most powerful knowledge source available to the inference engine. They represent the expert's knowledge, and they direct the actions of the expert system towards a desired goal. The general format of a rule is

if antecedent **then** consequent **endif.**

The antecedent is a logical comparison which evaluates to either true or false. The antecedent must be true for the consequent to be performed. The consequent consists of KES commands which contribute, or drive, the system toward a goal. For the assembly executive, the rules formulated require an intimate knowledge of the physical operations, potential system states, and capabilities of the various hardware. Rules have been defined for capturing information pertaining to tray transfer operations, and path segment selection for strut and panel installation/removal operations.

The path the robot arm travels from a rest position above the storage canister to the installation point in the structure is divided into segments or states. Figure 10 presents two rules that are used to determine the next segment (next_state) in the installation path for a strut. For this illustration the robot arm is poised above the supply canister awaiting direction to proceed to the grasp point of the canister. The current location of the robot arm (current_state) and the direction of the robot arm's motion (phase) determine the next segment in the robot's path. The robot's phase (either into or out of the structure) is determined from the current location of the robot (current_state), the current location of the strut (current_strut>where), and the task or goal specified by the operator (target_state). The current location of the robot is maintained in a database, and the location of the strut is held within the class member for that strut. To determine whether or not the consequence of the *state* rule is performed, the *phase* rule must be evaluated. The execution (called firing) of a rule often depends upon other rules being satisfied. It is this backward chaining technique that makes rules so powerful.

The strut installation path from the pickup point of the strut at the canister through the installation point at the structure and return requires 22 rules. These 22 simplified rules replaced approximately 850 lines of FORTRAN code. The total knowledge base currently contains 59 rules; twenty-two rules for determining strut assembly paths as previously indicated, twenty-two for panel paths, and fifteen for transferring trays from the supply canister to the storage canister and vice versa.

Demons-

Demons provide a method for event-driven inferencing within KES. Where rules actively seek additional information in an attempt to satisfy a specific goal, demons remain passive until an event occurs which initiates their execution. Adding event-driven inferencing (demons) to backward-chaining inferencing (rules) makes for a more dynamic expert system by providing a natural way of expressing some types of knowledge. Demons are useful for monitoring attributes for new or changed values in an attempt to modularize the procedural portions of the knowledge base.

A demon is composed of two parts; a guard and a body. A guard is similar to the antecedent of a rule, and contains conditional statements to be evaluated. The body contains a list of commands that KES executes sequentially. Assigning a new value to an attribute in the guard constitutes an event, causing all associated demons (ie. demons with that attribute in their guard) to be evaluated. If the guard evaluates to true, then KES executes the commands in the body of the demon. In the assembly executive knowledge base, when a value is assigned to an attribute in the consequent of certian rules, a demon is activated, initiating event-driven inferencing.

Suppose the *state* rule of figure 10 evaluates to true, and the next segment in the strut installation path is determined to be the canister grasp point (GP_CAN). The demon in figure 11 is used to generate the command strings necessary to move the robot arm to GP_CAN. Following the example, first some preliminary flags are set and the end effector conditions are checked. By assigning a value of true to the attribute *check scar* (a), another demon is activated which

makes sure the end effector is in the configuration necessary for making a safe approach to the canister. The value returned by the end effector is stored in the attribute *ee response*, which is examined before continuing (b). An uncorrectable error during the end effector operation would cause a roll back of the system to the last successful state (c). A successful return from the end effector allows the expert system to send a command to the processor associated with the robot arm to reset the force/torque sensor (d). Installation conditions for the current strut are ascertained (e) before the command to send to the robot is synthesized (f and g). The slot and tray numbers are appended to the base command string (h), and the command is sent (i). The assignment of true to the *send merlin* attribute constitutes an event which activates yet another demon. The send merlin demon sends the command and evaluates the robot response. If the device operated successfully, the current state is updated (j). The *message* command (j) is the means for sending the new value for the robot state to the database through the embedded interface. An unsuccessful robot operation results in a reverse (k and l).

A demon can change the value of the attribute that triggered its execution resulting in recursive behavior. The body of a demon can also determine the value of another attribute which itself may have associated demons. These demons can be triggered, invoking forward chaining. By blending both forward and backward chaining in a recursive environment, the assembly executive knowledge base has evolved into a concise and powerful mechanism for representing assembly knowledge.

Benefits

The concise representation afforded by the rule-based system reduced the lines of code significantly over the procedural (FORTRAN) version. A number of additional capabilities have been added to the system (panel operations, end effector changes, and machine vision), and the number of lines of code is still far below that of the original FORTRAN version. This reduction has lead to increased maintainability, and modifications and upgrades have been performed rapidly. The knowledge base is easier to debug and modify because the knowledge is separate from the algorithms and is readily accessible at run time.

This structural assembly project is relatively simple compared to many of the in-space check-out and servicing tasks currently being proposed. Expert system techniques have already proven to be mandatory for effective system management in ASAL. Such knowledge-based methodologies are a requirement for the timely development and maintenance of these types of complex systems.

RESEARCH OPPORTUNITIES

The overall goal of the ASAL research is to develop a complete integrated assembly system which incorporates on-line, automated planning and scheduling functions. The expert system executive described in this paper represents a first step in an evolution toward such advanced capabilities.

A baseline automated assembly system for space structures has been successful in assembling and disassembling a 102-member tetrahedral truss and demonstrating the utility of a supervised autonomy mode of operation. Complete assembly of the truss with the 12 attached panels using machine vision and the microprocessor controlled end effectors, all under the control of the expert system executive is being initiated. This test will demonstrate the capabilities of both the hardware and the software. In addition, performance data will be gathered which will help direct the evolution of the system. An attempt will be made to quantify error recovery actions taken by the operator with the goal of automating many error recovery procedures.

Currently when an error occurs, a menu of potential solutions is presented to the operator. The operator must then assess the error by visually verifying sensor data and select one or more options from an error recovery menu. By recording and studying the operators choices, the state of the system when the error occurred, the order in which error recovery actions are attempted, and the successful actions as well as failures, it is hoped that many processes can be automated. The final decision on error resolution will still rest with the operator, but a number of historically successful error recovery actions can be attempted before operator intervention is requested.

The enhancement with the largest software impact within ASAL will be the change over from the current system architecture (as seen in figure 6) to the highly distributed architecture as depicted in figure 12. Under this new architecture, all the devices will have their own processors, and will be controlled by an expert system scheduler. Maintaining separate devices for the individual processors will allow for concurrency among many assembly operations.

A number of advanced planners, each with their own knowledge base, are also included in the design of the new architecture. Knowledge bases will exist for: (1) a tray storage planner so that a fixed tray and slot assignment per strut will no longer be necessary, (2) a task planner for developing assembly scenarios based upon a definition of truss geometry and stiffness characteristics, (3) a path planner for determining a collision free path to the structure without having to rely on pre-determined approach points, and (4) a planner for combining necessary operations as a logical sequence and determining what actions can take place concurrently.

To manage the increased number of knowledge bases and individual processors, the KES software has been upgraded to an applications development tool known as the Strategic Networked Applications Platform (SNAP). SNAP supports the development of applications that operate in a distributed hardware environment. SNAP is made up of five components of pre-built software, the most important of which is the object model, containing the information driving the application. SNAP supports an object-oriented model of application data providing a direct mapping of the real world objects associated with the application to objects in the object model. Objects (classes) defined in the object model can be processed using either a rule-based knowledge source (backward chained rules), event-driven procedures (demons), or functions. Other components for building end-user interfaces using windows, mapping to permanent storage such as databases or files, integrating new or existing code, and communicating with other devices combine with the object model to make a complete application. Existing KES applications can be directly converted into SNAP compatible applications.

CONCLUDING REMARKS

The research conducted in ASAL has successfully demonstrated the viability of using robotic manipulators to automatically assemble and disassemble large truss structures. During the construction of a given structure, the system software assembly executive is responsible for making decisions about the actions to take, and the order in which to take them. To make informed decisions the

executive has to have access to all current system information, and the knowledge to evaluate that information in the light of the desired task. Due to the complexity of the software, continued implementation in traditional programming languages (ie. FORTRAN) became prohibitive. Traditional programming languages are not well suited for encapsulating the knowledge required for intricate assembly sequences. Preliminary investigations into the application of expert system technologies to perform the decision-making portions of the assembly software have been very encouraging.

Planned enhancements include implementation of a distributed architecture and several advanced planners. Multiple devices, each with their own processors, will be controlled by an expert system scheduler. The addition of a number of advanced planners, each with their own knowledge base, will make for a robust and reliable assembly system.

REFERENCES

1. Rhodes, Marvin D.; Will, Ralph W.; and Wise, Marion A.: *A Telerobotic System for Automated Assembly of Large Space Structures,* NASA TM-101581, March 1989.

2. Rhodes, Marvin D.; and Will, Ralph W.: *Automated Assembly of Large Space Structures,* 41st. International Astronautical Congress, Dresden, GDR, October 6-13, 1990.

3. Will, Ralph W.; Rhodes, Marvin D.; Doggett, William R.; Herstrom, Catherine L.; Grantham, Carolyn; Allen, Cheryl L.; Sydow, P. Daniel; Cooper, Eric G.; Quach, Cuong C.; and Wise, Marion A.: *An Automated Assembly System for Large Space Structures.* Intelligent Robotic Systems for Space Exploration. edited by Alan Desrochers, Academic Publishers, 1992.

4. Doggett, William R.; Rhodes, Marvin d.; Wise, Marion A.: and Armistead, Maurice F.: *A Smart End-Effector for Assembly of Space Truss Structures,* Fifth Annual Space Operations, Applications, and Research Symposium, Houston, TX, July 9-11, 1991.

5. Wu, K. Chauncey; Adams, Richard R.; and Rhodes, Marvin D.: *Analytical and Photogrammetric Characterization of a Planar Tetrahedral Truss.*, NASA TM-4231, 1990.

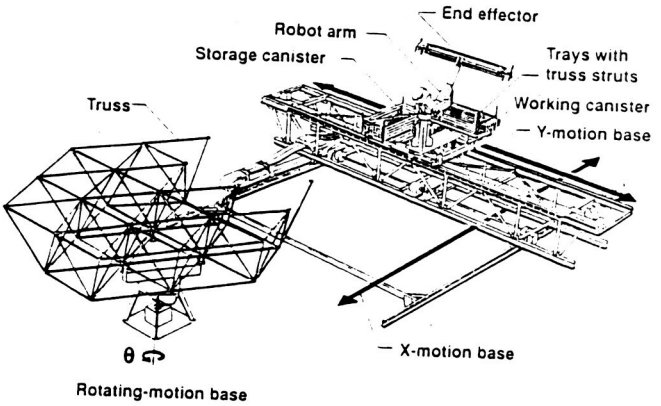
6. Kenner, Scott W.; Rhodes, Marvin D.; and Fichter, W.B.: *Component Count and Preliminary Assembly Considerations for Large Space Structures,* NASA TM-102604, Feburary 1990.

7. Sydow, P. Daniel ; and Cooper, Eric G.: *Development of a Machine Vision System for Automated Structural Assembly*, NASA TM-4366, May 1992.

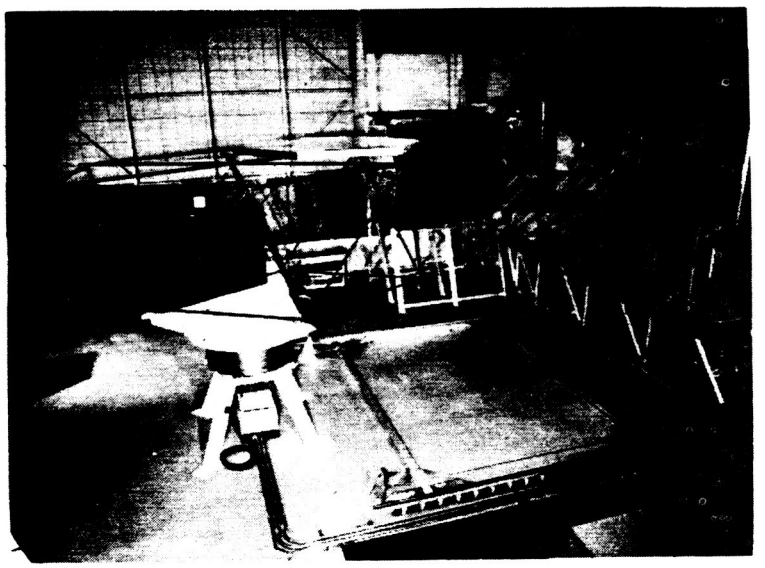
8. Herstrom, Catherine L.; Grantham, Carolyn; Allen, Cheryl L.; Doggett, William R.; and Will, Ralph W: *Software Design for Automated Assembly of Truss Structures,* NASA TP-3198, June 1992.

9. *Knowledge Base Author's Manual - KES PS.* Software Architecture & Engineering, Inc., c.1990.

10. *Knowledge Base Author's Reference Manual - KES PS.* Software Architecture & Engineering, Inc., c.1990.



(a) Schematic



(b) Photograph

Figure 1. Automated Structures Assembly Laboratory

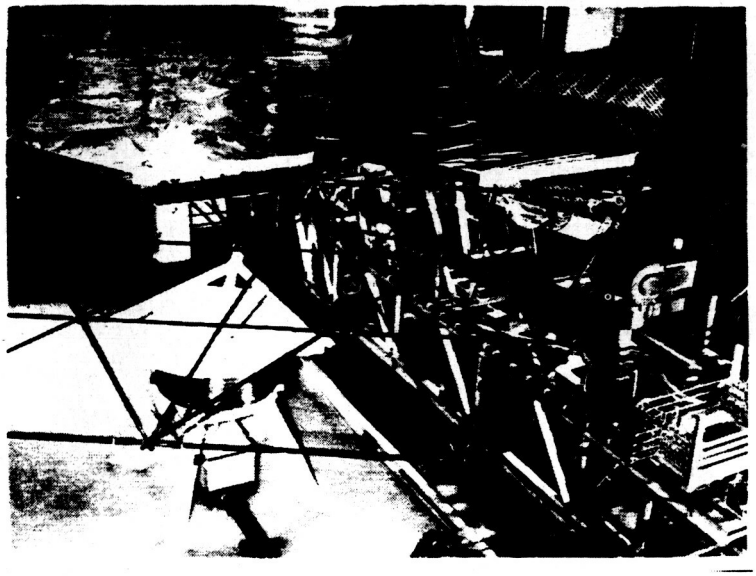
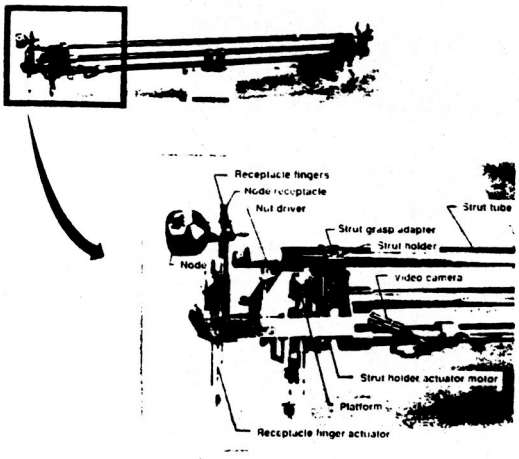
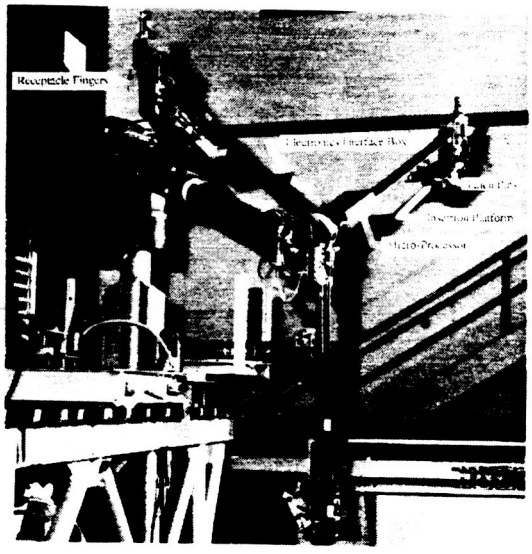


Figure 2. Tetrahedral Truss with Hexagonal Panels



(a) Strut End Effector



(b) Panel End Effector

Figure 3. ASAL End Effectors

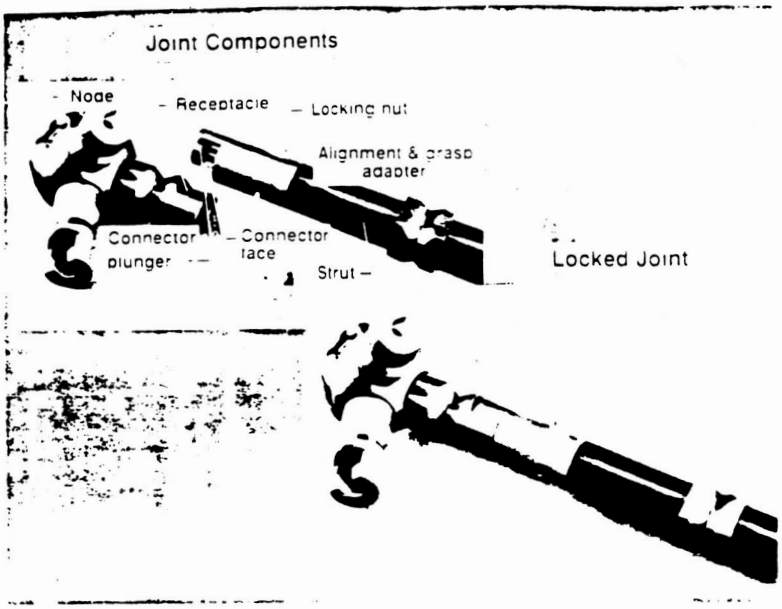


Figure 4. Strut/Node Joint Connector Hardware

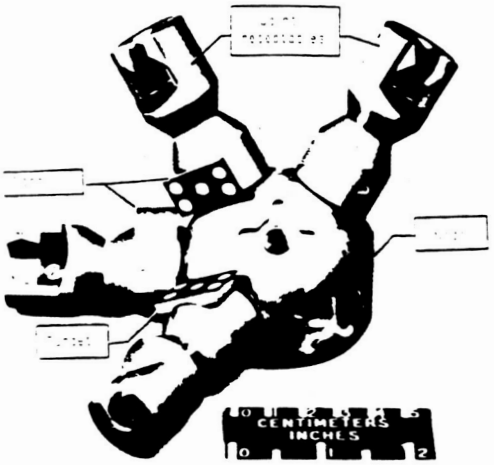


Figure 5. Truss Node with Joint Receptacle Targets

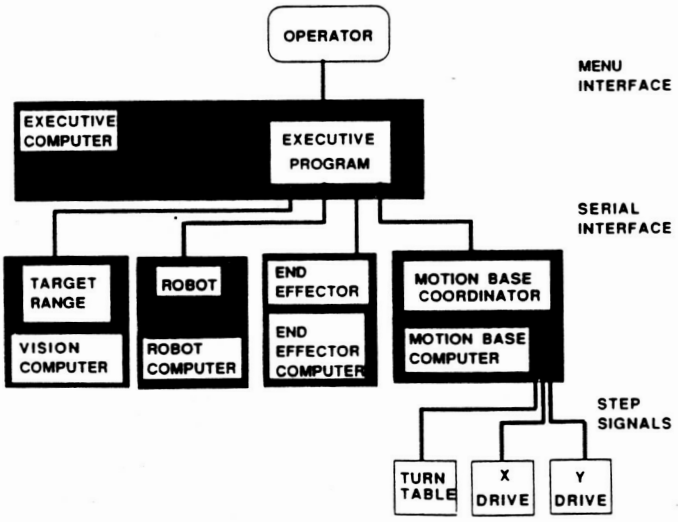


Figure 6. ASAL Computer Control System

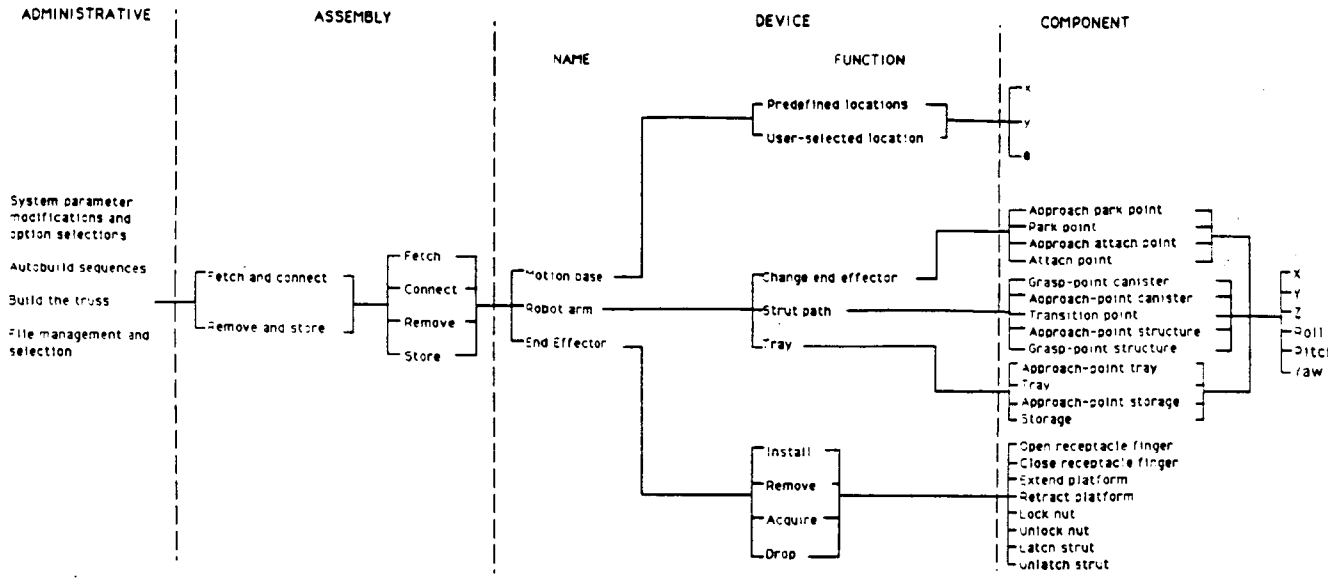


Figure 7. Software Design Layout

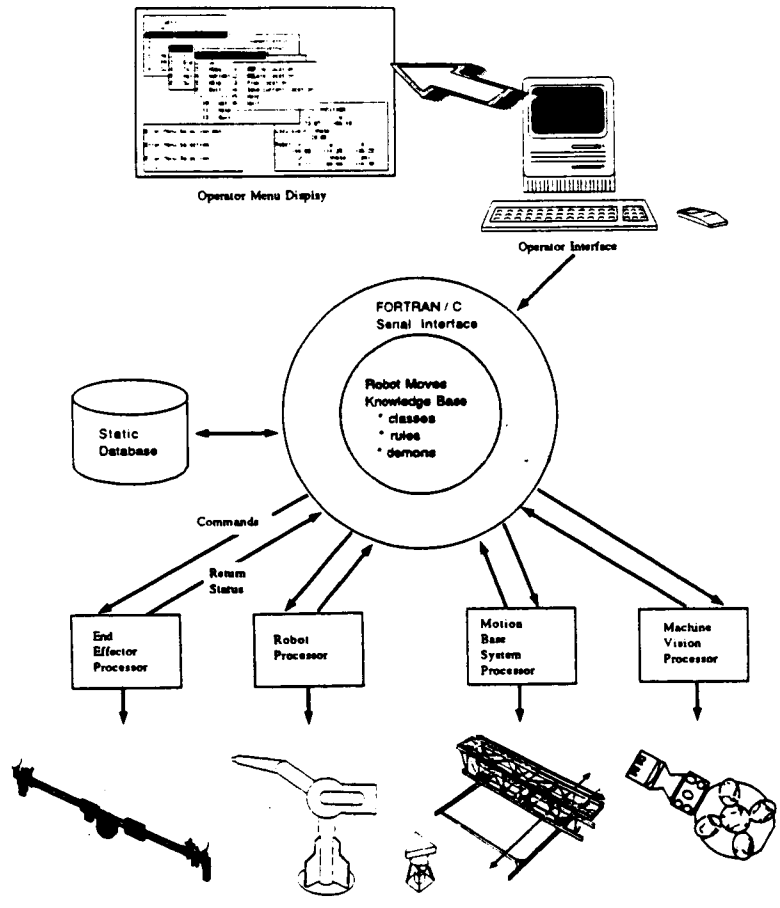


Figure 8. ASAL Software Architecture

```
Classes:

 STRUTS:
      attributes:
            OBSERVER NAME: str.
            ALTERNATE NAME: str.
            ROBOT NAME: str.
            TRAY: int.
            SLOT: int.
            NODE END: str.
            CAP END: str.
            WHERE: sgl (CANISTER, INSTALLED, ARM).
            FLIP: str.
            CAN_FLIP: truth.
            NODE DIRECT: str.
            MB_INDEX1: int.
            MB_INDEX2: int.
      %
endclass.
```

Figure 9. Class Defination for Struts

```
State:
 if
       current_state = AP_CAN* and
       phase = out
 then
       next_state = GP_CAN**
 endif.


Phase:
 if
       current_state = AP_CAN and
       target_state = GP_CAN and
       current_strut>where = CANISTER | ARM
 then
       phase = out
 endif.

*      AP_CAN : Canister approach point
**     GP_CAN: Canister grasp point
```

Figure 10. Example Rules for Strut Path Determination

```
State GP_CAN:
      when
            next_state = GP_CAN
      then
            reassert rule_flag = false.
            erase status_mode.
(a)         reassert check_scar = true.
(b)         if ee response = reversed then
(c)               reassert return = true.
            else        \ee response = worked
                  if ((init = false and restart = false) or override) and
                  status_mode = false then
(d)                     message "COMMAND$reset fts".
                  endif.
(e)               if current strut>CAN_FLIP then
(f)                     reassert tomerl = "GOTO GP_FLIP_CAN*"
                  else
(g)                     reassert tomerl = "GOTO GP_CAN*"
                  endif.
                  if determined (current strut) then
(h)                     reassert tomerl = combine(tomerl,current strut>SLOT,"*",
                                                  current strut>TRAY).
                  endif.
(i)               reassert send merlin = true.
                  if halt_op = false then
                        if robot success then
(j)                           message "UPDATE$charstate,GP_CAN".
                              reassert current_state = GP_CAN.
                        else  \return to calling state
                              if current strut>CAN_FLIP then
(k)                                 reassert tomerl = combine(
                                                "GOTO REV_GP_FLIP",
                                                current strut>SLOT,"*",
                                                current  strut>TRAY).
                                    reassert send merlin = true.
                              endif.
(l)                           reassert return = true.
                        endif.
                  endif.
            endif.
      endwhen.
```

Figure 11. Demon for Moving Robot to Canister Grasp Point

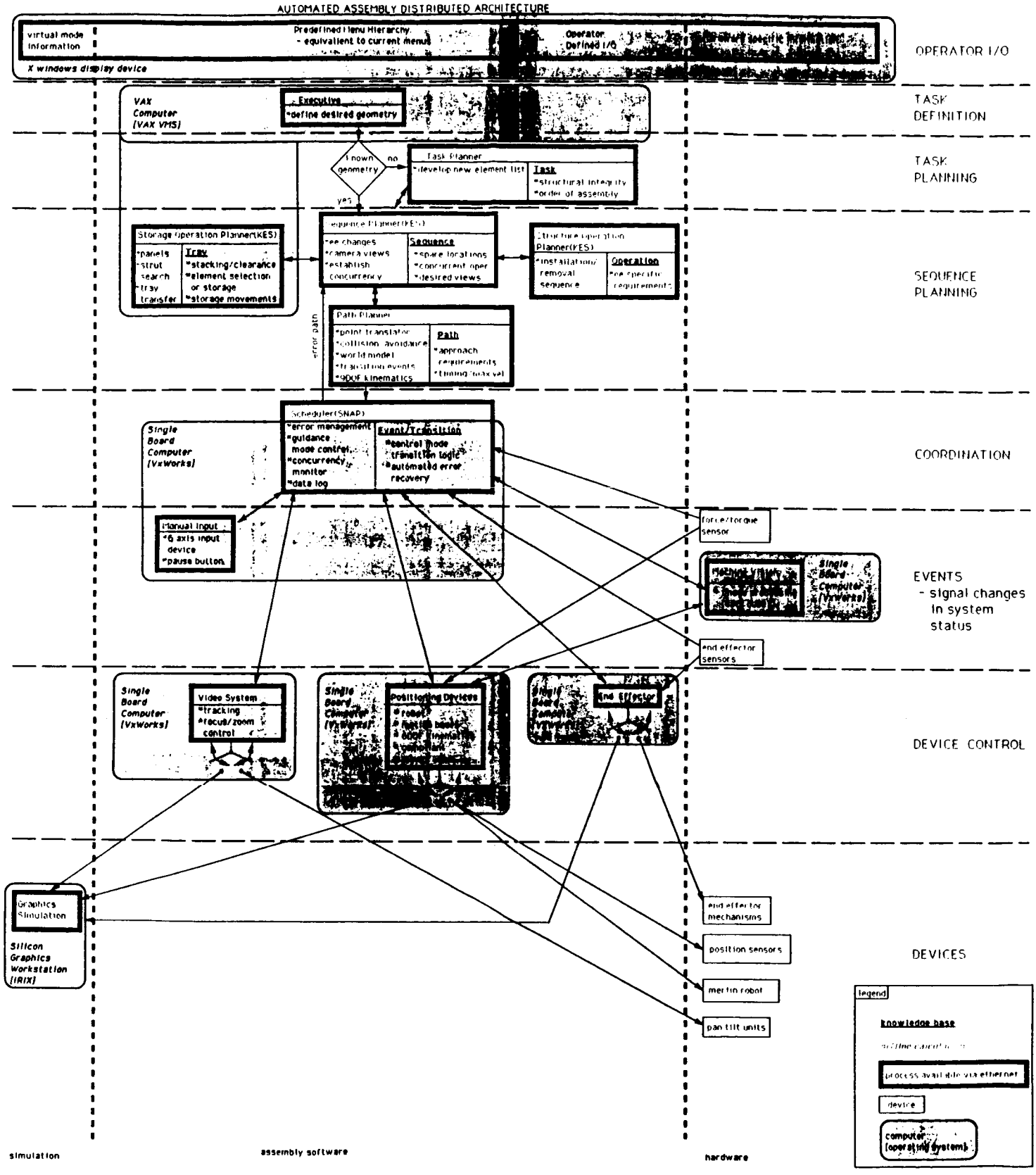


Figure 12. ASAL Distributed Architecture

Stanford Aerospace Robotics Laboratory Research Overview

W. L. Ballhaus * L. J. Alder † V. W. Chen ‡ W. C. Dickson §
M. A. Ullman ¶
Stanford University Aerospace Robotics Laboratory
Stanford, California 94305

Abstract

Over the last ten years, the Stanford Aerospace Robotics Laboratory (ARL) has developed a hardware facility in which a number of space robotics issues have been, and continue to be addressed. This paper reviews two of the current ARL research areas: navigation and control of free flying space robots, and modelling and control of extremely flexible space structures.

The ARL has designed and built several semi-autonomous free-flying robots that perform numerous tasks in a zero-gravity, drag-free, two-dimensional environment. It is envisioned that future generations of these robots will be part of a human-robot team, in which the robots will operate under the task-level commands of astronauts. To make this possible, the ARL has developed a graphical user interface (GUI) with an intuitive object-level motion-direction capability. Using this interface, the ARL has demonstrated autonomous navigation, intercept and capture of moving and spinning objects, object transport, multiple-robot cooperative manipulation, and simple assemblies from both free-flying and fixed bases.

The ARL has also built a number of experimental test beds on which the modelling and control of flexible manipulators has been studied. Early ARL experiments in this arena demonstrated for the first time the capability to control the end-point position of both single-link and multi-link flexible manipulators using end-point sensing. Building on these accomplishments, the ARL has been able to control payloads with unknown dynamics at the end of a flexible manipulator, and to achieve high-performance control of a multi-link flexible manipulator.

1 Experiments in Autonomous Navigation and Control of Multi-Manipulator, Free-Flying Space Robots

1.1 Introduction

Although space presents an exciting new frontier for science and manufacturing, it has proven to be a costly and dangerous place for humans. It is therefore an ideal environment for sophisticated robots capable of performing, as part of a human-robot team, tasks that currently require the active participation of astronauts.

As our presence in space expands, it will be increasingly important for robots to be capable of handling a variety of tasks ranging from routine inspection and maintenance to unforeseen servicing and repair work. Under the task-level guidance of astronauts, such tasks could be carried out by free-flying space robots equipped with sets of dexterous manipulators. These robots will need to be able to navigate to remote job sites, rendezvous with free-flying objects, perform servicing or assembly operations, and return to their base of operations.

In order to advance the underlying theory and technology necessary for the aforementioned robotic capabilities to be realized, the ARL has identified and addressed the problems associated with building and controlling autonomous

*Ph.D. Candidate, Department of Aeronautics and Astronautics.
†Ph.D. Candidate, Department of Aeronautics and Astronautics.
‡Ph.D. Candidate, Department of Electrical Engineering.
§Ph.D. Candidate, Department of Aeronautics and Astronautics.
¶Ph.D. Candidate, Department of Aeronautics and Astronautics.

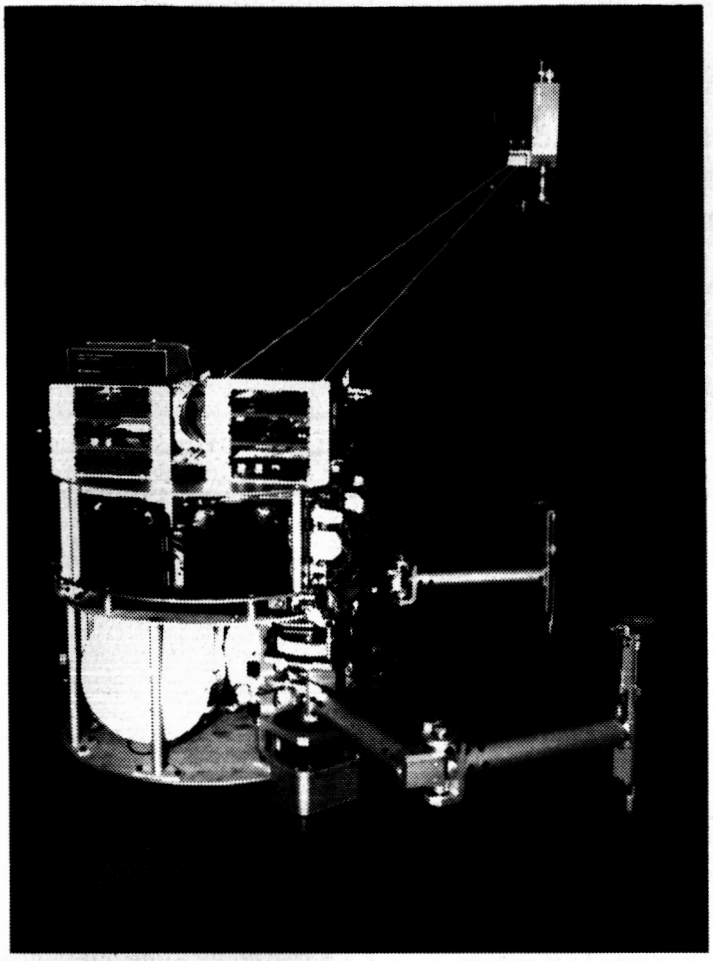


Figure 1: Stanford Multi-Manipulator Free-Flying Space Robot

This is a fully self-contained 2-D model of a free-flying space robot complete with on board gas, thrusters, electrical power, computers, camera, and manipulators. It exhibits nearly frictionless motion as it floats above a granite surface plate on a 0.005in thick cushion of air.

free-flying space robots. The objective of this research has been to demonstrate the ability to carry out complex tasks including acquisition, manipulation, and assembly of free-floating objects based on task-level commands. The approach has been to extend earlier ARL work in cooperative manipulation involving the use of fixed-base manipulators[1] to accommodate an actively mobile base thereby removing the workspace limitations inherent in fixed-base implementation.

1.2 Experimental Hardware

To test newly developed design methodologies and control strategies, the ARL has developed an experimental two-armed satellite robot shown in Figure 1. The robot uses an air cushion support system to achieve—in two dimensions—the drag-free, zero-g characteristics of space. The robot is a fully self-contained spacecraft possessing an on board gas supply for flotation and propulsion, rechargeable batteries for power, and on-board computers with sensing and driver electronics for navigation and control. Although the robot can function autonomously, its computers can also communicate with a network of workstations via a new wireless LAN. An on-board camera provides optical endpoint and target sensing while an overhead global vision system facilitates robot navigation and target tracking.[1] The robot "floats" on a $9' x 12'$ granite surface plate with a drag-to-weight ratio of about 10^{-4} and gravity induced accelerations below $10^{-5}g$—a very good approximation to the actual conditions of space. A more detailed description of the space robot is given in [2].

[1] The global vision system serves as a convenient laboratory surrogate for a tracking system such as GPS that could be used for this purpose in space.

1.3 Controller Architecture and Graphical User Interface

The controller architecture consists of a three-level hierarchy composed of a stateless remote graphical user interface, a high-level strategic controller, and a low-level dynamic controller based on an operational space computed torque formulation.

The graphical user interface (See figure 2) runs on a Sun Workstation and allows an operator to send high level commands to the robot by selecting icons and clicking on buttons.

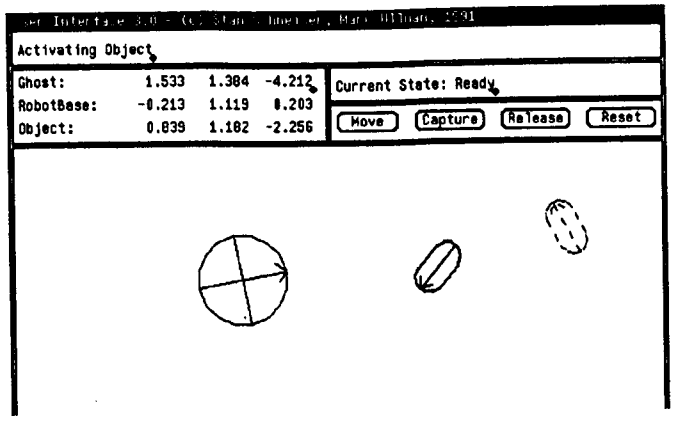


Figure 2: **Typical View of the User Interface**

The graphical user interface provides "point and click" operation of the robot and allows the operator to control and monitor all operations remotely. Here a capture and move operation is underway.

The high-level strategic controller is based on a sophisticated finite state machine. It accepts commands from the remote user interface and reconfigures the low-level dynamic controller to carry out desired actions. A thorough discussion of the strategic controller is beyond the scope of this paper and can be found in [2].

1.4 Object Rendezvous and Capture

The execution of useful work in space requires the ability to simultaneously control both robot base and manipulator motions. In general, rendezvousing with and capturing a free-flying object requires controlling both manipulator and base body positions to follow *coordinated* intercept trajectories. Global navigation and control (or "gross motion" control) of a space robot therefore poses a set of interesting and unique challenges. These differ fundamentally from both the typical satellite positioning/attitude control problem and the case of a free-floating space robot with an uncontrolled base.

The robot can be commanded to capture and retrieve an object via the graphical user interface described in the section *Controller Architecture and Graphical User Interface.* Figure 3 shows the time history of one such rendezvous. The object floats on an air bearing supplied by a battery-powered aquarium air pump. The object can be sent across the granite table in a random direction with a rotation rate as high as 20 revolutions per minute. Since the object is of comparatively low mass ($\sim 1kg$), the robot follows a straight line path to intercept it. Upon grasping the object by inserting its "peg-in-the-hole"-style grippers, the robot brings the object smoothly to rest (in the robot's reference frame). It can then stow the object and transport it to some new location where it can release it.

1.5 Nonlinear Adaptive Control of a Free-Flying Space Robot

Adaptive control for robots is useful in several important, common situations: 1) When there is poor or no knowledge of the payloads, 2) When there are inaccurate models of the robot, 3) When there are changes in the environment. While a robust, nonadaptive, controller may provide the same protections as an adaptive controller in these situations, it typically does so with substantially reduced performance. The added complexity of an adaptive controller wins back that lost performance.

The most obvious situation in which to use adaptive control is for handling payloads that have unknown or poorly known physical properties—for example, when handling damaged satellites where the nature and extent of the damage

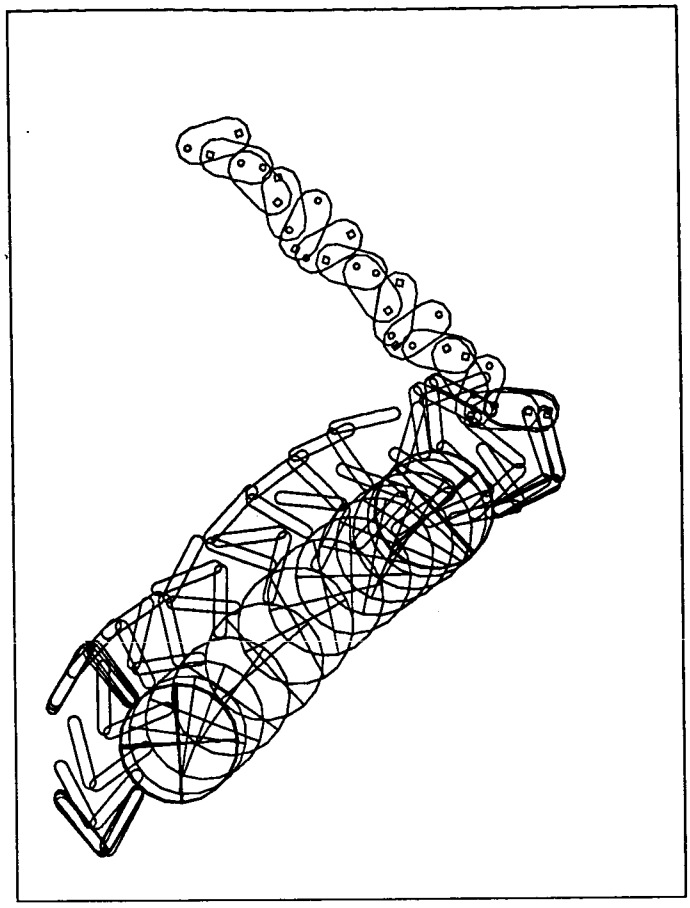


Figure 3: **Time history of Object Rendezvous and Capture**

This "time-lapse" plot of experimental data shows the motion of a spinning object and the path the robot executed in order to intercept and capture it. The frame rate is 0.5Hz and this figure shows about 30 seconds of elapsed time.

is unknown. More generally, this capability relieves astronauts of the duty to inform the robot of the detailed physical properties of each payload the robot is to handle. While a comprehensive parts database can relieve much of this responsibility, adaptive control provides protection in cases when the database is not completely accurate or is lacking.

Adaptive control also eases the basic controller design process. There are typically many aspects of the robot itself that are either poorly modelled or not modelled. It is difficult to develop accurate models for robots. While lengths and masses can be measured relatively accurately, effective center of mass locations and moments of inertia of individual links after being incorporated into the robotic system are estimates at best. Other poorly characterized effects, such as joint friction, and forces caused by wiring harnesses, contribute to an inaccurate model. By providing appropriate adjustable parameters to the controller, adaptive control can adapt to these uncertainties to render their effects unimportant.

This research has generated an adaptive control framework that is very general and easily extensible to even larger, more complex systems than the free-flying robot with cooperating manipulators for which it was developed [3]. The contributions made by this research include:

- Development of a general adaptive control framework—the adaptive *task*-space framework—that is capable of providing full adaptation to a free-flying space robot with two cooperating manipulators in all modes of operation.
- Extension and generalization of a joint-space, nonlinear, adaptive control algorithm, based on inverse-dynamics, to control in the *task* space, which represents a broader class of control inputs, including, but not limited to, cooperative *object* control, as well as endpoint control and joint control.

- Formulation of the *system concatenation* approach for efficient, incremental generation of system models for multiple, interacting systems. *System concatenation* takes full advantage of models already developed for each manipulator or robot subsystem to minimize the effort in deriving the total system models used for adaptation.
- Full integration of the new adaptive control algorithm into a hierarchical control architecture that includes a graphical user interface and a finite state table programming environment.
- Experimental verification of the new adaptive controller, in the hierarchical control environment, on a free-flying space robot model.
- Development of the "Point-Grabber II" vision system that, together with software drivers developed in ARL, is capable of tracking bright spots at 60 Hz with better than 1/20 pixel resolution.

1.6 Cooperative Object Manipulation by Free-Flying Robot Teams

Free-flying space robots could perform many of the dangerous and expensive extra-vehicular activity (EVA) tasks presently requiring humans, such as the assembly and maintenance of structures and off-board platforms. Certain missions may require two or more robots to work as a team to cooperatively manipulate large objects such as satellites or structural elements. Situations requiring this capability include the initial installation or assembly of objects, or the retrieval of objects for repair or replacement. Manipulating flexible or multi-body objects is another task which may require a team of robots—a single robot may not be able to adequately control the object's internal degrees of freedom.

With this motivation in mind, the overall objective of this research was to identify and address the central dynamics and control issues relating to object manipulation by free-flying robot teams [4]. In achieving this goal, this research made the following contributions to automatic control and robotics:

- The concept of a team manager was developed as a mechanism for directing the activities of multiple independent robots into a cooperative team effort.
- Task-level direction was extended for use with free-flying robot teams for the first time. Once the user specifies a desired object position, the robot team determines and executes the control commands required for proper manipulation.
- A *Hybrid-Dynamics* formulation of equations of motion was developed that determines the mixed set of system accelerations and controls that is consistent with a specified complementary mixture of accelerations and controls. This formulation provided the basis for a robot control algorithm that combines discrete-valued base thrusters with proportional arm motors to produce precise manipulator endpoint accelerations.
- A dynamic modelling method has been developed for producing a system model directly from subsystem descriptions. This method is based on the solid foundation of constrained-system theory and takes advantage of simple kinematic relationships to merge subsystem descriptions into a full system model with no need for subsystem decomposition.
- Laboratory experiments have successfully demonstrated a team of free-flying robots capturing and manipulating a large, freely moving object. In these experiments, a human user indicates a desired object location and orientation through a graphical user interface. The robots then capture and so position the object with no additional input required from the user.

2 Control of Flexible Space Manipulators

2.1 Introduction

Many current and future space missions either require now or will require the assistance of large scale manipulator systems. For example, the Shuttle Remote Manipulator System (SRMS) has aided in the deployment, maneuvering, and retrieval of satellites and orbiter payloads. Future space robotic tasks will include spacecraft inspection and maintenance, transportation of payloads about space structures, and docking maneuvers.

To maximize the workspace of space robotic manipulators, the links of these robots tend to be very long in length. Furthermore, due to the cost of boosting mass into orbit, lightweight manipulator designs are most desirable. The

resulting large, lightweight space manipulators contain low-frequency inherent structural flexibility which increases the difficulty of achieving high-performance control. As a result, our ability to model and control these flexible manipulators will determine how efficiently they can be used to perform various tasks.

Large flexible space robots like the SRMS are currently controlled by astronauts using joy sticks to control the position of each manipulator joint. To avoid exciting the low-frequency modes of the SRMS, the astronauts move the SRMS very slowly so that the end-point speed remains typically below 0.06 m/sec and 0.6 m/sec in its fully loaded and unloaded configurations respectively.

Previous ARL research has demonstrated that the performance of large lightweight manipulators can be significantly increased by improving the system controller. Through the use of direct end-point sensing, the end-point position can be used in a feedback control strategy to achieve accurate control of the manipulator end-point [5] [6].

Current ARL research in the control of flexible space manipulators extends this earlier work to control payloads with unknown dynamics at the end of a flexible manipulator, and to achieve high-performance control of a multi-link flexible manipulator.

2.2 Control of a Flexible Robotic Manipulator with Unkown Payload Dynamics

Space-based robots such as the SRMS and the proposed Space Station Remote Manipulator System (SSRMS) have been and will be essential elements of future space exploration. Further, the payloads manipulated by these large, flexible robots may themselves include unknown internal dynamics. Examples include sloshing fuel and/or flexible appendages (e.g. vibrating solar panels on a small satellite). If the payload dynamics are not accounted for in the control design, degraded performance or instability are possible.

The existing body of control that has been developed for flexible space robotic manipulators is insufficient to achieve high-performance control of such complex configurations. Collocated controllers are insensitive to payload dynamics but are low performance [7]. High-performance control has been achieved by using non-collocated end-point control [7], but the configurations studied all involved payloads that were well modelled by a tip inertia matrix only. Further, it has been shown that these high performance non-collocated controllers are sensitive to the mass and inertia of the tip [8]. It has also been demonstrated that a controller that is tuned for a particular tip mass may in fact be unstable for a different tip mass [8]. The sensitivity to an unknown tip mass has been accounted for successfully using adaptive endpoint control [9].

The goal of this research is to develop control techniques that provide precise, high bandwidth end-point control of flexible manipulators and are also able to damp any internal oscillations of the payload. The internal dynamics of the payload will not be known a priori. Furthermore, it is assumed that it will not be practical to outfit the payload with sensors that measure the internal state of the payload. The sensory input for controlling the robot-payload system will be based on the robot system sensors only. [2]

The control approach that has been developed and demonstrated experimentally in this research is based on extensions to the self-tuning regulator solution of adaptive control (see [10] for details). The extensions yield a feasible, real-time control that potentially can be implemented in future space robotic systems. High performance control is merged with an innovative identification algorithm in a self-tuning regulator approach. The identification of the payload is done using recently developed subspace fitting techniques. These techniques allow real-time determination of the order of the payload dynamics. Sufficient excitation problems are addressed by performing the identification closed loop.

Figure 4 shows a schematic diagram of the hardware. The experiment represents a large space-based manipulator holding a payload that has internal vibrating dynamics. The robot arm is a flexible beam which moves in the horizontal plane. At one end of the beam, there is a motor, and on the other end there is the payload which is a pad that floats on an air bearing. The pad and air bearing prevent out-of-plane vibrations. Mounted on the pad is a pendulum that represents the dynamics of the payload. Floating the pad on a smooth granite table simulates the zero-g environment of space in one dimension.

The pendulum inside the payload can be held mechanically so that it will not vibrate, or it can be set free to oscillate. The length of the pendulum can be modified to simulate an unknown frequency of oscillation (i.e. one that might be encountered in a sloshing fuel tank). The damping ratio of the pendulum is on the order of 0.5%. When released form a 45° initial condition, it takes about one minute to damp within $\pm 5°$ of its steady-state value. Reference [11] has a more detailed description of the hardware including mass properties.

The effect of having unmodelled payload dynamics on the closed-loop performance of the system can be seen by examining the time response shown in Figure 5. Figure 5 shows the time response of an end-point-based LQG controller

[2] This includes vision sensing.

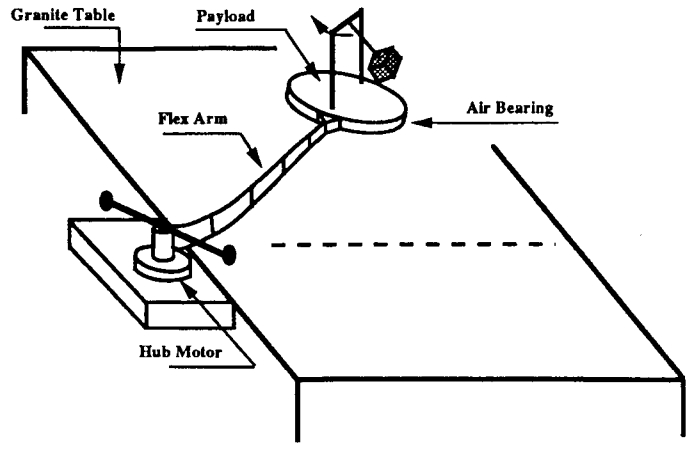


Figure 4: **Hardware Schematic**

This figure shows a schematic of the single link flexible manipulator with a dynamic payload.

that neglects the dynamics of the payload (i.e. the payload is considered to be a rigid body). This inadequate time response verifies that the payload dynamics cannot be neglected when using end-point control.

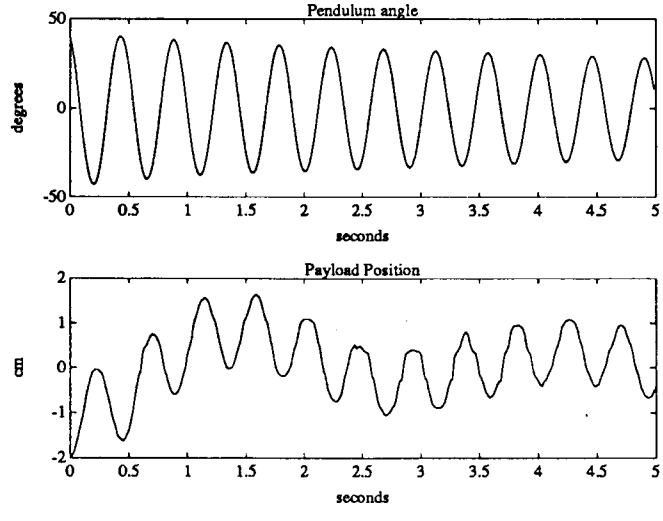


Figure 5: **Time Response**

This figure shows the time response of an end-point-based LQG controller that neglects the dynamics of the payload. This inadequate time response verifies that the payload dynamics cannot be neglected when using end-point control.

The experimental results of the adaptive controller are shown in Figure 6. The pendulum is mechanically held at a 45^o initial condition so that the payload is initially a rigid body. While the controller is regulating the payload position, the pendulum is released. For the first 2.2 seconds, the pendulum is being held at 45 degrees and the payload is regulated to zero. At 2.2 seconds, the pendulum is released resulting in motion of the payload. For the next two seconds, the pendulum is damping very slowly with a damping ratio of less the 0.5%. During this time, the identification algorithm detects the frequency of the pendulum, and at four seconds into the run, it swaps in a controller that accounts for the payload dynamics. Three seconds later, the pendulum is damped to within five degrees and the arm is within two millimeters of its desired position.

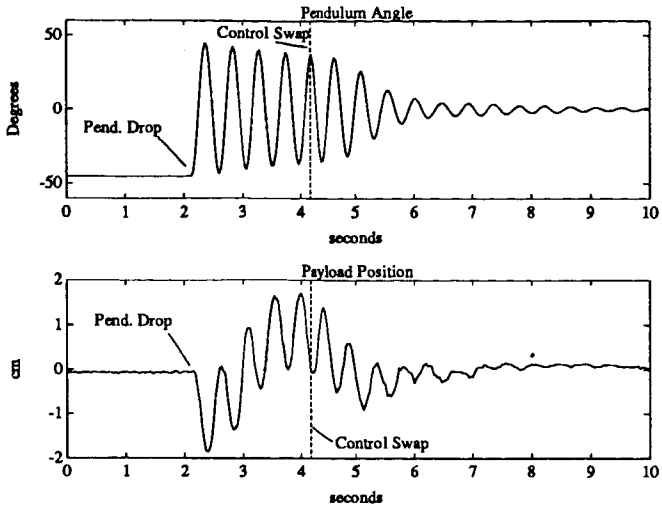


Figure 6: **Time Response**

This figure shows the time response of the adaptive controller. The identification algorithm detects the frequency of the pendulum, and then swaps in a controller that accounts for the payload dynamics.

2.3 High-Performance Control of a Multi-Link Flexible Manipulator

This research focuses on the problem of repositioning quickly and accurately a multi-link flexible space structure within a specified time (see [12] for details). It is assumed that only the initial and final states of the structure and the final time of the slew are given. A quadratic performance index is formulated which weights the value of the state at the terminal time, and the integral of the square of the control effort expended to that time. Minimizing this performance index with respect to the control effort results in a terminal controller with time-varying feedback gains.

For many applications, the motion of the end-point of a structure may be arbitrary, as long as the structure reaches a desired configuration within a specified time. For example, suppose a large flexible space robot was initially performing a task in one portion of its workspace, and then was commanded to perform a task at another location. In this case, assuming that the workspace is clear of obstacles, the trajectory that the endpoint follows in moving from its original working location to the commanded position is not of concern, as long as the manipulator reaches its desired destination, or repositions itself, quickly. Similarly, the transportation of rigid payloads from one location to another is an additional application where only the initial and final positions and the time duration of the slew are of importance.

There are various approaches that can be taken to reposition a flexible structure. For example, when a desired trajectory is known, it is possible to calculate an open-loop set of control inputs which would result in the desired end-point motion. Another method of repositioning a flexible structure or manipulator involves using full state feedback based on optimally determined, time-invariant (TI) feedback gains derived from a linear-quadratic cost function. This approach is fully described in [13] and [6]. Although these methods have been shown to perform quite well, they require a specified trajectory from which the control inputs are calculated. The terminal controller design discussed in this research automatically generates, based on the minimization of a linear-quadratic performance index, the optimal state history and the corresponding feedback gains necessary to enforce this motion. Terminal controllers have the added advantage of being computationally inexpensive to calculate (allowing for on-line controller design), and are independent of the initial and final states of the system. Furthermore, the time duration of the repositioning is the only slew parameter upon which the terminal controller design is dependent. As a result, the same set of controller gains can be used for point-to-point slews having equal specified final times.

The Stanford Multi-Link Flexible Manipulator, shown in Figure 7, consists of a two-link flexible manipulator which operates on air cushions in the horizontal plane of a 1.2 m by 2.4 m granite table. Each of the flexible links is 0.52 m in length, and consists of an aluminum beam (cross section 1 mm by 38.1 mm) with discrete masses evenly spaced along its length. The discrete masses, termed "Mass Intensifiers", increase the overall beam mass without changing its flexural rigidity. These Mass Intensifiers also lower the natural frequencies of vibration. The flexible links exhibit significant bending in the horizontal plane due to their narrow cross section and orientation. Air cushion supports at the elbow

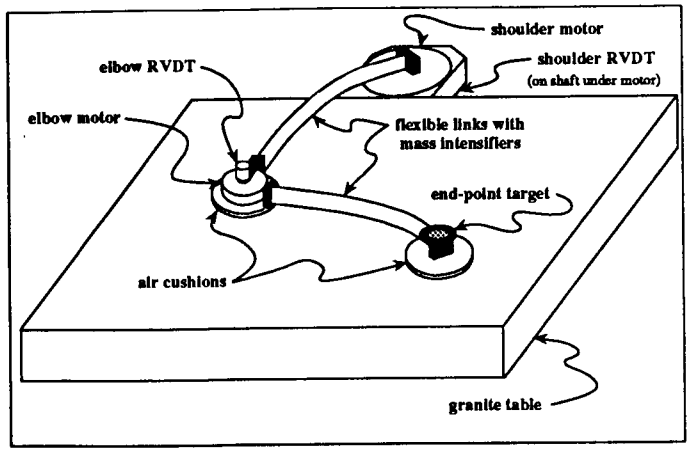


Figure 7: **Schematic of the Stanford Multi-Link Flexible Manipulator**

Shown in this figure is the Stanford Multi-Link Flexible Manipulator along with the various components of the experimental apparatus.

and end-point provide torsional stiffness. In addition to exaggerating the link flexibility, the structure was designed to be lightly damped. As a result, damping must be provided to the system through active control. The shoulder motor, mounted on the side of the granite table, can provide a peak torque of 5.43 N-m. The elbow motor, mounted on the elbow air-cushion pad, can provide a peak torque of 1.06 N-m. Both actuators are direct-drive, DC limited-angle torquers. Rotary variable differential transformers (RVDT's) are located at each of the motor shafts and provide joint angle measurements. A vision sensor, fully described in [1], is available for end-point measurements. It consists of a CCD television camera that tracks a special variable reflectivity target located at the manipulator end-point. The vision system has the capability to track multiple targets at a sample rate of 60 Hz with a resolution of approximately 1 mm over the roughly 1.5 m^2 workspace [6].

Figures 8 and 9 show a comparison between experimental and simulated responses for the terminal controller for a final slew time of $t_f = 2.5$ seconds. From Figure 8 it is evident that both the shoulder and elbow motor position histories

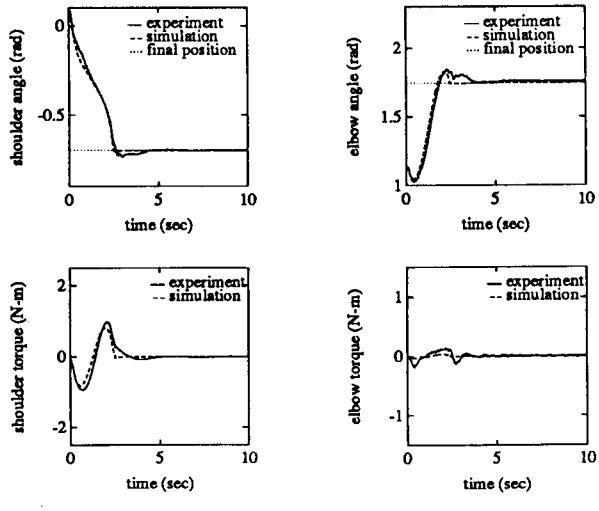


Figure 8: **Terminal Controller Experimental Response for $t_f = 2.5$ seconds**

Comparison of experimental and simulated position and torque histories for the shoulder and elbow motors.

correspond quite well with the simulated results, with the exception that the experimental responses show slightly more

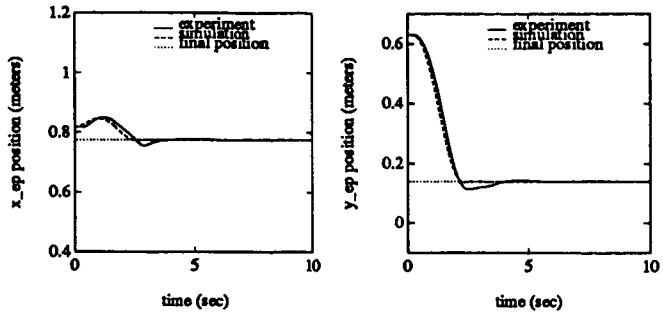


Figure 9: **Experimental End-Point Motion for $t_f = 2.5$ seconds**

Experimental and simulated end-point response for the terminal controller.

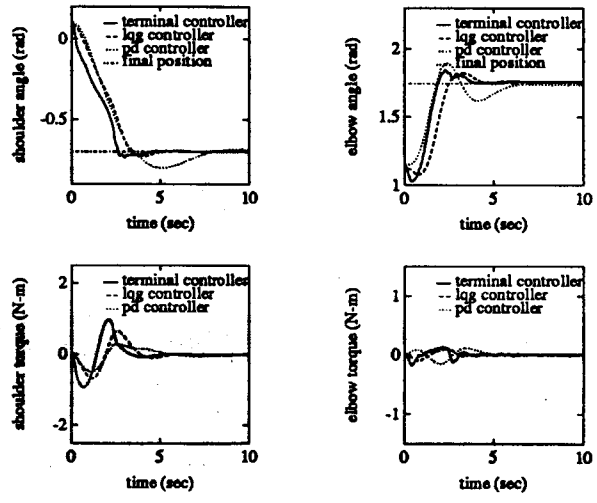


Figure 10: **Controller Comparison for $t_f = 2.5$ seconds**

A comparison of experimental position and torque histories for the terminal controller, an endpoint based LQG controller, and a PD controller.

overshoot in both cases. The endpoint position response shown in Figure 9 again shows excellent agreement between experiment and simulation.

Figure 10 gives a comparison of the terminal controller to an LQG end-point-based controller and a PD controller. The response of the PD controller is by far the worst, as expected, in that it exhibits excessive overshoot and a large settling time. Considering the end-point response, Figure 11 illustrates that the terminal controller shows significantly shorter rise times and settling times than the LQG controller, with comparable overshoot [12].

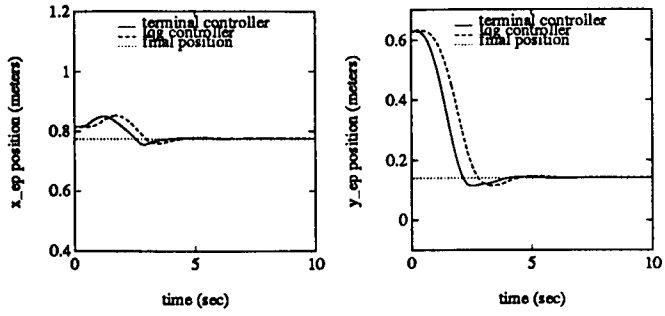


Figure 11: Controller Comparison of End-Point Motion for $t_f = 2.5$ seconds

Experimental and simulated endpoint responses for the terminal controller and an end-point based LQG controller.

Acknowledgements

The research reported here has been funded by the National Aeronautics and Space Administration and the Air Force Office of Scientific Research. The authors would like to thank Professor Robert H. Cannon Jr. and all of the students and technical staff at the Stanford Aerospace Robotics Laboratory for their support, assistance, and suggestions.

References

[1] S. Schneider. *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators.* PhD thesis, Stanford University, Stanford, CA 94305, September 1989. Also published as SUDAAR 586.

[2] M. A. Ullman. *Experiments in Autonomous Navigation and Control of Multi-Manipulator Free-Flying Space Robots.* PhD thesis, Stanford University, Stanford, CA 94305, (February) 1992. To be published.

[3] Vincent W. Chen. *Experiments in Adaptive Control of Multiple Cooperating Manipulators on a Free-Flying Space Robot.* PhD thesis, Stanford University, Stanford, CA 94305, (September) 1992. To be published.

[4] William C. Dickson. *Experiments in Cooperative Object Manipulation by Free-Flying Robot Teams.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, (October) 1992. To be Published.

[5] Eric Schmitz. *Experiments on the End-Point Position Control of a Very Flexible One-Link Manipulator.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, June 1985. Also published as SUDAAR 548.

[6] Celia M. Oakley. *Experiments in Modelling and End-Point Control of Two-Link Flexible Manipulators.* PhD thesis, Stanford University, Department of Mechanical Engineering, Stanford, CA 94305, April 1991.

[7] Robert H. Cannon, Jr. and Eric Schmitz. Initial experiments on the end-point control of a flexible one-link robot. *The International Journal of Robotics Research*, 3(3):62–75, Fall 1984.

[8] Daniel M. Rovner and Robert H. Cannon, Jr. Experiments toward on-line identification and control of a very flexible one-link manipulator. *International Journal of Robotics Research*, 6(4):3–19, Winter 1987.

[9] Daniel M. Rovner and Gene F. Franklin. Experiments in load-adaptive control of a very flexible one-link manipulator. *Automatica*, 24(4):541–548, July 1988.

[10] Lawrence Alder. *Experiments in Adaptive Control of a Flexible Link Robotic Manipulator with Unknown Dynamic Payloads.* PhD thesis, Stanford University, Stanford, CA 94305, (January) 1992. To be published.

[11] L. J. Alder and S. M. Rock. Control of a flexible robotic manipulator with unknown payload dynamics: Initial experiments. In *Proceedings of the ASME Winter Annual Meeting*, pages 61–66, Atlanta GA, December 1991. ASME.

[12] W. L. Ballhaus, S. M. Rock, and A. E. Bryson, Jr. Optimal control of a two-link flexible manipulator using time-varying controller gains: Initial experiments. In *Proceedings of the American Astronomical Society Guidance and Control Conference*, Keystone, CO, February 1992.

[13] C. M. Oakley and R. H. Cannon, Jr. Anatomy of an experimental two-link flexible manipulator under end-point control. In *Proceedings of the IEEE Conference on Decision and Control*, Honolulu, HI, December 1990.

The Technology Base for Agile Manufacturing

R. C. Brost, D. R. Strip, and P. J. Eicker*
Intelligent Systems and Robotics Center
Sandia National Laboratories
Albuquerque, NM 87185-5800

The effective use of information is a critical problem faced by manufacturing organizations that must respond quickly to market changes. As product runs become shorter, rapid and efficient development of product manufacturing facilities becomes crucial to commercial success. Effective information utilization is a key element to successfully meeting these requirements. This paper reviews opportunities for developing technical solutions to information utilization problems within a manufacturing enterprise, and outlines a research agenda for solving these problems.

1 Introduction

Agile manufacturing is being heralded as the key to successful competition in today's marketplace, supplanting traditional mass production techniques. As always, the successful economic competitor is the one who produces the right product at the right price. What has changed is the marketplace — consumers no longer want a generic product at the lowest price. The increasing reality of the world as a single market is forcing suppliers to deal with more varied demands, niche markets, and most importantly, more competitors. Agility in manufacturing is the the ability to respond to small production quantities, short development cycles and product lifetimes, and increased product variety. The manufacturer who successfully adopts these skills will be more responsive to changing customer desires and will be rewarded with a larger market share.

It is our view that although there are important process development issues, success in agile manufacturing will center on effectively utilizing information in the manufacturing organization. As product cycles and production quantities shrink, product costs become increasingly dominated by non-recurring engineering costs — the upfront engineering. Furthermore, efficient information control is a critical contributor to determining time-to-market, another increasingly important issue in today's marketplace. The successful producer is the one who recognizes the importance in developing, analyzing, and utilizing information.

*This work was supported by the U.S. Department of Energy under contract DE-AC04-76DP00789.

The manufacturer that ties its future productivity to the utilization of information also opens an opportunity to free itself from current resource-based growth constraints. As long as competitiveness is closely tied to physical goods, a mature industry finds its fate tied to changes in resource availability or to the incremental improvements of a mature technology. By linking its future to information utilization, the agile enterprise benefits from the inherently unlimited nature of an abstract commodity, information. Further, information technology is currently improving at a very high rate, allowing information to be processed at rapidly declining costs.

Sandia's Intelligent Systems and Robotics Center (ISRC) has created a program in Information-Driven Manufacturing to focus efforts on the effective control and utilization of information in the manufacturing environment. We have identified automatic planning and programming, and sensor- and model-based control as key technologies required to overcome the impediments to agility inherent in current generation manufacturing equipment. We have chosen assembly of electromechanical components to motivate our research in agile manufacturing. This subset of manufacturing tasks is important both for its major role in manufacturing, and because many of the key issues to be resolved in electro-mechanical assembly are representative of analogous problems elsewhere in the manufacturing process.

In classic economic tradition, we will draw upon the example of the Acme Widget Company to build a scenario for agile manufacturing. Through this example we will illustrate the path of information flow in the manufacturing process and explore how better utilization of that information translates into greater agility. We will also identify the important research issues that require resolution to create the highly agile enterprise.

2 The Acme Widget Company

Acme Widget is a world-leader in the design and manufacture of — widgets. Market research has identified significant consumer interest in widgets with a Super Turbo feature. Fortunately, the technology required to produce Super Turbo Widgets has been developed by researchers in Acme's basement. Now Acme's problem is to manufacture the new Super Turbo Widgets and bring

them to the marketplace before their competitor, Vile Enterprises. We will follow the design and production of this new product with an eye toward where information is generated and utilized.

Acme researchers have produced a prototype Super Turbo Widget, but this prototype must be refined into a commercial product design and then manufactured. The engineering department develops a candidate design, and the drawings are sent to the prototyping shop. Based on tests with the early units, engineering makes a number of design changes to remove slight performance glitches, and new prototypes are built. After several such cycles, the design is frozen and sent to production engineering.

The basic technology of the Super Turbo Widget is the same as the ordinary widget, so production engineering's primary task is to develop tooling and set up the new production line. Production engineering develops an assembly line plan, and notices that several inverting stations are required that seem extraneous. By inserting screws A12–A16 from the opposite direction, several of these inverting stations may be omitted, reducing manufacturing costs. Production engineering sends the drawings back to design engineering and requests the design change.

Design engineering implements the requested change, which requires cutting new dies for some of the stamped parts. Production engineering then resumes work. An initial batch of parts is fabricated, and assembly line construction begins. Initial trials reveal that the hole spacing on part X3Y is too tight for the high-speed insertion machines, requiring either a design change or a significant cost penalty along with reduced production rate. This hole spacing is not critical to widget performance, so the design is again modified by design engineering.

Finally the unit goes into volume production. During the initial production runs, the workstation for part G42Z jams 15% of the time, causing spoilage that interrupts production and eradicates profits. Apparently there is a slight difference between the injection-molded production version of part G42Z and the machined prototypes used in the initial workstation trials. Production engineers study the situation, and are eventually able to adjust and modify the workstation tooling to prevent the problem.

At last Acme is producing Super Turbo Widgets without a hitch. Unfortunately, Vile Enterprises introduced its new Extra Turbo Whatsit two months earlier, and has already sold 20,000 units. It is unclear whether Acme will ever be able to gain a significant market share in the wake of Vile's head start.

3 Analysis of the Scenario

The scenario demonstrates a basic cycle consisting of design → fabricate → test → repeat. In our example we had an "inner loop" between engineering and prototyping that we traversed several times. There is another inner loop in the production engineering area. Finally, there is an outer loop containing these two inner loops. There are three basic methods that we can apply to shorten the time to get to market — fewer times around each loop, concurrent execution of the loops, and faster execution of the steps inside each loop. The first and last methods also lead to reduced non-recurring engineering costs.

Acme could have exploited all three of these methods by using information effectively. For example, production engineering's initial trials showed that modifying the X3Y hole spacing could reduce manufacturing costs; if this information about the limitations of the manufacturing equipment was discovered at design time, an entire cycle between design and production engineering could have been avoided. Further, if design engineering had released interim design information, then production engineering could have begun assembly line development concurrently. Finally, better analysis tools could have helped both design and production engineering identify and resolve problems more rapidly, thus shortening the time required for each step.

Inadequate use of information was central to Acme's problems. The most significant problems were related to information *content* rather than information *flow*. For example, the failure of the G42Z workstation resulted from subtleties in the interaction between the workstation and the parts it manipulated. This workstation design was sensitive to differences between machined and injection-molded parts, but this information was not known until production began, at which point the workstation began to fail. This is an example where critical information was simply not available until late in the design/production cycle.

Despite the close similarity of the manufacturing processes for the ordinary and Super Turbo Widgets, the scenario shows how inadequate information utilization during design and manufacturing development can substantially affect the eventual profitability of the product.

4 A Research Agenda for Agile Manufacturing

The Acme widget scenario showed two opportunities for using information to increase manufacturing agility: by allowing work to proceed concurrently, and by providing early detection of problems. There is also a third approach: by using information on-line to provide process capabilities not available otherwise. These new process capabilities may eliminate problems and reduce the need for complex design analysis.

In the following sections we outline an agenda of research required to support the information-based agile enterprise. The outline provides the broad brushstrokes of what will be a long-term effort by many participants. Within Sandia's Intelligent Systems and Robotics Center we have created the Information-Driven Manufacturing Project to develop technology to support each of

these approaches, ultimately in concert. The following sections will review these approaches, the research problems they entail, and some of Sandia's efforts toward their solution.

4.1 Allowing Concurrent Work

One method of reducing the time required to take a product to market is to perform product design and manufacturing development concurrently. This requires good communication between design and manufacturing groups, and frequent exchange of design information between groups. Information exchange adds overhead to the efforts of both groups; if this overhead becomes too large, then the benefits of concurrent engineering are lost.

Successfully implementing concurrent engineering in a manufacturing organization involves significant human factors and management issues. However, there are technical challenges as well. These include the development of efficient communication techniques that minimize the overhead associated with concurrent engineering. Supporting this communication electronically could produce substantial benefits, but some difficult problems must be confronted. For example, distributed design representations must be developed that will allow a design to be modified by many designers spread throughout a manufacturing organization, while preventing one designer from inadvertently corrupting another designer's work. Further, the design representation must allow each designer to view relevant aspects of the design; for example, one designer may wish to check for part interference during the expected motions of a mechanism, while a second designer may focus on the mechanism's hydraulic control system. At the same time these activities are proceeding, a production engineer may be seeking ways to reduce the complexity of assembling the mechanism. Each of these analysis problems requires numerous analysis-specific details describing the design. If these views are not seamlessly integrated, then concurrent engineering is compromised because the implications of design changes are more difficult to transmit between groups.

We can break the problem of developing concurrent engineering tools into two basic components: representation and communication. The representation component requires that the design database support each designer's view of the design problem. In the example of the previous paragraph, the representation must support analysis of kinematic linkages, geometric sweeping operations for interference checking, a model of hydraulic performance, and the geometric and physical information needed for assembly analysis. The communication component requires software that allows multiple designers to effectively access the data simultaneously. This software must facilitate efficient communication of design details and intent, and prevent conflicts between design modifications.

Past work has addressed many of these issues. Representations have been developed for geometric analysis, including kinematic pairs and sweeping operations [Leu *et al.* 1986; Hoffmann 1989; Joskowicz 1989; Rossignac and Requicha 1986]. Special-purpose representations have also been developed for modeling hydraulic systems and other design aspects. However, representations that support manufacturing process analysis require further development; these must be completed before an integrated multiple-view design representation may be developed. There has also been work addressing the communication component of the problem [Kannapan and Marshek 1991]. Past work in computer networks, transaction-based databases, and multiple-user concurrency and version control all shed light on the issues that must be addressed to successfully implement a concurrent engineering system.

Given the state of the art, a sound approach to developing effective concurrent engineering systems is to develop the necessary specific representations to support analysis of manufacturing processes, integrate these and other representations to produce an integrated multiple-view design representation, and develop a system that will allow several users to work simultaneously with a design using the integrated representation. The near-term research goals of the Sandia ISRC will address the first of these problems by developing the representations required to support early problem detection and on-line control of manufacturing processes. Integrating these representations will be addressed in the longer term.

4.2 Early Problem Detection

One of the key lessons of the Acme scenario is that it becomes increasingly expensive to resolve problems as the development process proceeds. Further, the later a problem is discovered, the greater the delay that results from solving the problem. Thus, early detection of problems in the product or its manufacturing facility could reduce non-recurring engineering costs while simultaneously bringing the product more quickly to the market. In other cases these techniques may show that manufacturing costs will erode profits, allowing a manufacturer to cancel a project before the sunk costs become too great.

Problems may occur in a variety of places. The product may fail to perform its desired function reliably. Basic part fabrication processes may produce high scrap rates. Assembly machines may jam or improperly assemble parts. To maximize the agility of the manufacturing enterprise, we would like to detect problems as early as possible in each of these areas. In this section, we will focus on problems that occur in the assembly stage. This is due to the inherent difficulty of predicting these problems, their prevalence in manufacturing applications, and their relative insensitivity to the fabrication technology used to produce the component parts. The goal of the research agenda outlined here is to develop

analysis techniques that can combine the information inherent in the design model with information regarding manufacturing capabilities to predict manufacturing problems.

Assembly problems may occur on a macroscopic or microscopic level. On the macroscopic level, the intrinsic product design may force additional manipulation and assembly operations that increase manufacturing costs (such as the extra inverting stations needed in our Acme Widget scenario). Analyzing assembly strategies early in the design process may suggest relatively simple design changes that may dramatically reduce the number of required assembly operations and yield substantial manufacturing cost savings. This benefit could be maximized by performing an assembly analysis frequently during the design process, but this is impeded by the inherent difficulty of a detailed assembly analysis. The number of possible assembly sequences grows exponentially with the number of parts, and complex feasibility constraints must be evaluated to discriminate between feasible and infeasible assembly plans. Further, these constraints depend on details of the assembly line and its tooling, which may be modified to improve assembly efficiency. These complications discourage frequent assembly analysis; tools to aid this analysis would allow manufacturing organizations to simplify product assembly operations before design changes become prohibitively expensive.

The problem of automatic assembly sequence analysis has been studied for several years. The 1992 IEEE Conference on Robotics and Automation sponsored a workshop on this topic; the notes to this workshop provide a snapshot of the state of the art [Lee *et al.* 1992]. More accessible references include [Fahlman 1974; Lieberman and Wesley 1977; Boothroyd and Dewhurst 1986; Wolter 1988; Strip and Maciejewski 1990; Defazio *et al.* 1990; Homem de Mello and Lee 1991]. These efforts have produced several prototype assembly planning and analysis systems, and have also identified a variety of open problems that must be solved to support general assembly analysis. These include developing planners that accept geometric CAD models as input, implementing search procedures that effectively cope with the combinatorial explosion of design alternatives, developing methods for combining geometric and physical process constraints in the assembly analysis, and implementing reconfigurable procedures for generating output process specifications and machine-executable code. The Sandia ISRC will continue to address these problems in future work.

Assembly problems may also occur at a finer level of detail. In addition to laying out the simplest possible sequence of assembly operations, manufacturing engineers must implement assembly lines that reliably carry out each operation. This is at once critical and difficult. Due to the serial nature of assembly lines, the reliability of the entire line hinges on the reliability of each individual operation. Assuring this reliability is very difficult to do in advance, since part-handling operations often involve subtle mechanical processes that are hard to understand and predict. The Acme example contains an instance of this — the inability to reliably feed part G42Z because of variations induced in the fabrication stage. Currently a manufacturing engineer is forced to employ trial-and-error methods after parts and tooling have been fabricated, which delays assembly line implementation and leads to late (expensive) design changes. Tools that help identify reliability problems early on would help manufacturing organizations avoid these delays and their associated cost.

Research on issues at this microscopic scale are generally referred to as *fine motion planning* [Lozano-Pérez *et al.* 1984]. This research strives to simultaneously consider task geometry, physics, and uncertainty to automatically analyze and construct robust assembly tasks. Much of the early work in this area studied issues related to specific assembly tasks, such as peg-in-hole assembly [Whitney 1982; Erdmann 1986; Donald 1988; Strip 1989; Caine *et al.* 1989]. Other work has addressed tasks more closely related to part feeding and presentation [Erdmann and Mason 1986; Peshkin and Sanderson 1988; Goldberg *et al.* 1991; Boothroyd 1992; Brost 1992; Schimmels and Peshkin 1992]. These results have improved our ability to automatically analyze assembly tasks, but much work remains. Open problems include the development of robust analysis procedures that may be applied to a variety of tasks, definition of practical models of task uncertainty, and calibration of the physical accuracy of the analysis output. Future ISRC research will address these problems.

The above discussion separates the macroscopic and microscopic aspects of an assembly problem. The macroscopic analysis addresses the assembly sequence and overall assembly plan, while the microscopic analysis addresses the details of each individual assembly operation. In reality, these issues are coupled — the choice of assembly sequence can simplify or complicate the required assembly operations, and details of each assembly operation can affect the feasibility of a given assembly sequence. An ultimate goal of our research in information-driven manufacturing is to produce an integrated assembly analysis tool that supports early problem detection by simultaneously considering both macroscopic and microscopic aspects of an assembly problem.

4.3 On-Line Process Control

The previous sections have described two methods for using information to reduce the time to implement a desired manufacturing line — by supporting concurrent work and early problem detection. Both methods have the benefit that, given a fixed set of problems to solve in order to implement a desired assembly line, they accelerate the process of solving these problems.

A third approach is to reduce the number of problems that must be solved by increasing the capabilities of the assembly process itself. One method is to use information to make on-line process adjustments at production

time. For example, consider the problem of mating two parts. A standard approach is to place one part precisely in a fixture, and then attach the second part using a placement device that expects the first part to be repeatably placed. Implementing this strategy requires the development of a fixture and fixture-placement operation that places the first part in a highly-repeatable position; the details of this operation must be carefully evaluated by a manufacturing engineer. An alternative would be to place the first part in a simple clamping device, measure the part's true position, and modify the second part's insertion motion in response to the measured position. If the sensing/motion modification device is easy to reprogram for the new task, then this approach reduces the effort required to develop the assembly line because the need for a detailed repeatable fixture analysis is eliminated. Additional cost savings arise because a part-specific custom fixture does not need to be constructed, and part tolerances relevant only to the fixturing operation may be relaxed.

On-line control of manufacturing processes can also produce cost savings by allowing changes to the technology used to manufacture parts. For example, sometimes a weld between two parts must be ground flush with the surrounding surface to eliminate stress risers on an assembly that is subject to high strain. The exact surface position is not critical, but the surface smoothness is critical. Precise grinding machines may be used to smooth out the weld to the required tolerance, but these machines also require a precise surface position and shape, thereby imposing additional tolerance specifications that result purely from manufacturing considerations. If on-line sensing is used to sense the surface position and adjust the grinding operation accordingly, then these additional tolerance specifications are no longer required. This may allow the parts to be cast instead of machined, yielding substantial cost savings.

These examples illustrate ways to employ the use of information during process execution, thereby simplifying the up-front information analysis required. Adequately using information on-line requires good sensors, means for interpreting sensor data, methods for controlling actuation devices based on the interpreted sensor results, and a simple method for re-programming the sensing/control device to execute a new task.

The use of on-line sensing to compensate for process variations has been applied to a variety of manufacturing processes. For example, active force control has been applied to compensate for part and fixturing inaccuracies during assembly operations [Whitney 1977]. Force control has also been used in conjunction with high-resolution capacitive sensors to allow tight-tolerance finishing operations on metal parts in significantly less time than competing technologies [Selleck and Loucks 1990]. Machine vision has been used to locate parts that are difficult to feed mechanically [Regalbuto 1991]. The challenge in each of these cases is to make the on-line sensing/control system flexible and easy to re-program. Future research at the ISRC will continue to develop sensors, control systems, and user interfaces to support flexible on-line process control.

5 Conclusion

We have seen that effective information utilization is a critical issue in agile manufacturing. Although there are numerous issues in process and tool development that will significantly impact agility in production, the most important changes will come from shifting our attention from the hardware in a plant to the information used to design and manufacture a new product. It is only through good information utilization that we can achieve better productivity in the non-recurring engineering costs, which are becoming increasingly important as product life cycles shorten. In addition, effective information utilization will shorten product development time, which is essential as the number and variety of competitors grows. On the factory floor, information-based control will change the way in which machines and processes work. As more of the process versatility becomes embedded in real-time controllers, we will be able to add previously unimagined capabilities to smart machines at low marginal cost.

Information-driven manufacturing methods use information to efficiently implement a successful manufacturing facility. In this paper we have identified an agenda of research topics whose resolution is necessary to make these methods a reality. The outlined research program will require a dedicated effort by many researchers from a variety of organizations. Like many others throughout the world, we in the ISRC are working on pieces of the puzzle. We recognize the critical importance of collaborating with others ranging from leading edge research organizations to down-in-the-trenches production engineers. We invite these collaborations as we continue to develop tools to support information-driven production.

References

[Boothroyd and Dewhurst 1986] G. Boothroyd and P. Dewhurst. *Product Design for Assembly.* Boothroyd Dewhurst Inc., Wakefield, RI, 1986.

[Boothroyd 1992] G. Boothroyd. *Assembly Automation and Product Design.* Marcel Dekker, New York, 1992.

[Brost 1992] R. C. Brost. Dynamic analysis of planar manipulation tasks. In *Proceedings, IEEE International Conference on Robotics and Automation,* pages 2247–2254, May 1992.

[Caine *et al.* 1989] M. E. Caine, T. Lozano-Pérez, and W. P. Seering. Assembly strategies for chamferless parts. In *Proceedings, IEEE International Conference on Robotics and Automation,* pages 472–477, May 1989.

[Defazio *et al.* 1990] T. L. Defazio, A. D. Edall, R. E. Gustavson, J. A. Hernandez, P. M. Hutchins, H. W. Leung, S. C. Luby, R. W. Metzinger, J. L. Nevins, K. K. Tung, and D. E. Whitney. A prototype for feature-based design for

assembly. Technical Report CSDL-P-2917, Charles Stark Draper Laboratory, July 1990.

[Donald 1988] B. R. Donald. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intelligence*, 37:223–271, 1988.

[Erdmann and Mason 1986] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. In *Proceedings, IEEE International Conference on Robotics and Automation*, pages 1569–1574, April 1986.

[Erdmann 1986] M. A. Erdmann. Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1):19–45, Spring 1986.

[Fahlman 1974] S. E. Fahlman. A planning system for robot construction tasks. *Artificial Intelligence*, 5:1–49, 1974.

[Goldberg *et al.* 1991] K. Y. Goldberg, M. T. Mason, and M. A. Erdmann. Generating stochastic plans for a programmable parts feeder. In *Proceedings, IEEE International Conference on Robotics and Automation*, pages 352–359, April 1991.

[Hoffmann 1989] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann, San Mateo, California, 1989.

[Homem de Mello and Lee 1991] L. S. Homem de Mello and S. Lee, editors. *Computer-Aided Mechanical Assembly Planning.* Kluwer Academic Publishers, Norwell, Massachusetts, 1991.

[Joskowicz 1989] L. Joskowicz. Simplification and abstraction of kinematic behaviors. In *Proceedings, 11th International Joint Conference on Artificial Intelligence*, 1989.

[Kannapan and Marshek 1991] S. M. Kannapan and K. M. Marshek. A schema for negotiation between intelligent design agents in concurrent engineering. In *Proceedings IFIP Intelligent CAD.* Elsevier, 1991.

[Lee *et al.* 1992] C. S. G. Lee, S. Lee, and A. C. Sanderson, editors. *The First Workshop on Assembly Planning: Theory and Implementation.* IEEE Robotics and Automation Society, May 1992.

[Leu *et al.* 1986] M. C. Leu, S. H. Park, and K. K. Wang. Geometric representation of translational swept volumes and its applications. *Journal of Engineering for Industry*, 108:113–119, May 1986.

[Lieberman and Wesley 1977] L. I. Lieberman and M. A. Wesley. Autopass: An automatic programming system for computer controlled mechanical assembly. *IBM Journal of Research and Development*, 21(4), July 1977.

[Lozano-Pérez *et al.* 1984] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, Spring 1984.

[Peshkin and Sanderson 1988] M. A. Peshkin and A. C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4(5):524–531, October 1988.

[Regalbuto 1991] M. Regalbuto. Vision-guided mechanical assembly. Videotape, Adept Technology, San Jose, CA, April 1991.

[Rossignac and Requicha 1986] J. R. Rossignac and A. A. G. Requicha. Offsetting operations in solid modeling. *Computer Aided Geometric Design*, 3:129–148, 1986.

[Schimmels and Peshkin 1992] J. M. Schimmels and M. A. Peshkin. Admittance matrix design for force-guided assembly. *IEEE Transactions on Robotics and Automation*, 8(2):213–227, April 1992.

[Selleck and Loucks 1990] C. B. Selleck and C. S. Loucks. A system for automated edge finishing. In *Proceedings of the IEEE International Conference on Systems Engineering*, August 1990.

[Strip and Maciejewski 1990] D. R. Strip and A. A. Maciejewski. Archimedes: An experiment in automating mechanical assembly. In *Proceedings, 11th International Conference on Assembly Automation*, November 1990.

[Strip 1989] D. R. Strip. A passive mechanism for insertion of convex pegs. In *Proceedings, IEEE International Conference on Robotics and Automation*, pages 242–248, May 1989.

[Whitney 1977] D. E. Whitney. Force feedback control of manipulator fine motions. *Journal of Dynamic Systems, Measurement, and Control*, pages 91–97, June 1977.

[Whitney 1982] D. E. Whitney. Quasi-static assembly of compliantly supported rigid parts. *Journal of Dynamic Systems, Measurement, and Control*, 104:65–77, March 1982. Reprinted in M. Brady, *et al.*, editors, *Robot Motion: Planning and Control*, MIT Press, Cambridge, Massachusetts, 1982.

[Wolter 1988] J. D. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly.* PhD thesis, University of Michigan Department of Computer, Information and Control Engineering, September 1988.

Sensor Fusion for Assured Vision in Space Applications

Dr. Marie-France Collin*, Dr. Kumar Krishen
NASA/Johnson Space Center
Code IA4 - Houston, TX 77058

Abstract

By using emittance and reflectance radiation models, the effects of angle of observation, polarization, and spectral content are analyzed to characterize the geometrical and physical properties—reflectivity, emissivity, orientation, dielectric properties, and roughness—of a sensed surface. Based on this analysis, the use of microwave, infrared, and optical sensing is investigated to assure the perception of surfaces on a typical lunar outpost. Also, the concept of employing several sensors on a lunar outpost is explored. An approach for efficient hardware implementation of the fused sensor systems is discussed.

Introduction

Human presence on the lunar surface for extended periods of time (for up to 2 years at the beginning of the next millennium) will require extensive supporting capabilities including habitat modules, power generation modules, operational control modules, and life support modules. Emplacement of this evolutionary lunar base will require preliminary robotic missions such as surface exploration or mining for construction purposes. Because of specific illumination conditions in space and mission requirements, achieving these operations—either automated or teleoperated—requires advanced sensing technologies to assure perception of the lunar scene at any time and in any location by a vision system.

*Marie-France Collin is on leave of absence from ITMI, BP 177, 61 Chemin du Vieux Chêne, 38244 Meylan Cedex, France.

The scene perception and interpretation capabilities of the vision system being designed at the NASA/Johnson Space Center with the collaboration of ITMI, France, will be based on physical models that underlie the reflection and emission radiations phenomena. These physical models take into account the relationship between environmental illumination (which can be active in the presence of radar sensors or passive in the presence of thermal or visible sensors), surface parameters, and perceived data. Physical models will be used jointly with fuzzy logic techniques to perform fusion of the multisensor data and to interpret the physical and geometrical properties of the sensed surfaces. The perception system architecture is represented by the following scheme (fig. 1), which shows sensor selection and fusion modules.

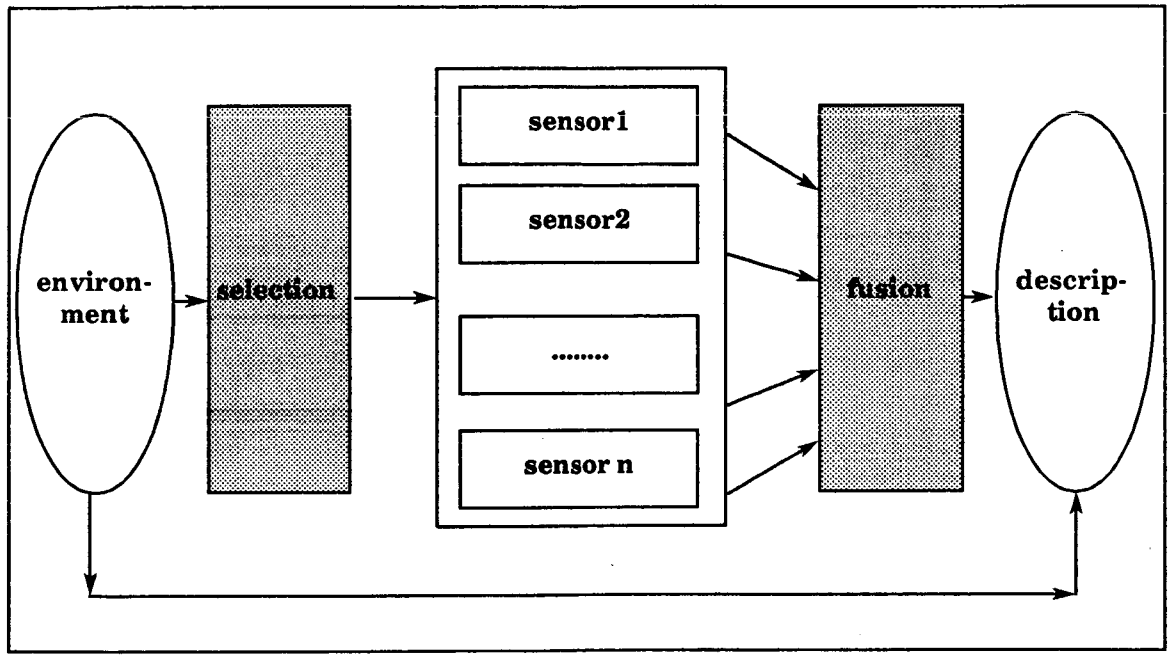


Figure 1. Perception system architecture.

This paper first presents the reflectance and emittance models on which sensor selection and sensor fusion are based. These models allow us to understand the effect of surface parameters on the response of multiple sensing devices. The constitutive key parameters are roughness, the dielectric constant, orientation, temperature, and emissivity of the surface. These surface parameters are needed for scene perception and interpretation in the context of planetary operations. The second part of the paper, which is based on reflectance and emittance model analysis, presents the effect of these

key parameters on sensor responses for different sensors. The third part of the paper deals with a method being developed to assure the perception and interpretation of the scene for space operations.

Physics of perception

In the remote sensing field, perception models have been used extensively to characterize surfaces for Earth observation purposes [1],[2],[3],[4]. The models presented here are issued from this domain. The following sections will present a survey on the most commonly used theoretical models for scattering and emission mechanisms.

Scattering models

Energy reflected off a surface and received by a remote sensing device is related to a scattering coefficient, σ_{pq}, that is dependent on surface physical properties. The subscript pq indicates that the received field is p-polarized and the transmitted field is q-polarized. The most common values for p and q are horizontal and vertical polarizations. To obtain a numerical solution of the scattering coefficients, σ_{pq}, several models have been developed that depend on the frequency range of illumination and on surface geometry. By making assumptions on the scattering mechanisms, it is then possible to get relatively simple numerical solutions for the scattered coefficients.

According to the Kirchoff approximation, a scattered field can be estimated using the Fresnel reflection coefficients R_v and R_h for vertical and horizontal polarizations. As shown in the following expressions [5], these coefficients depend on the electrical properties of the surface and on the incident angle:

$$R_h = [\mu \cos\theta - (\mu\varepsilon - \sin^2\theta)^{1/2}]/[\mu \cos\theta + (\mu\varepsilon - \sin^2\theta)^{1/2}]$$

$$R_v = [\varepsilon \cos\theta - (\mu\varepsilon - \sin^2\theta)^{1/2}]/[\varepsilon \cos\theta + (\mu\varepsilon - \sin^2\theta)^{1/2}]$$

Using the Kirchoff approximations, numerical simplification leads to the scattered coefficients estimation first derived by Beckmann-Spizzichino [7].

$$\sigma_{pq} = \sigma_{1pq} + \sigma_{2pq} + \sigma_{3pq}$$

σ_{1pq} is the specular reflection term, and σ_{2pq} and σ_{3pq} are due to the surface roughness and slope effects, respectively. Figure 2 presents a typical backscattering response as a function of the incidence angle θ using different values of the standard deviation of the surface heights σ (representing the surface roughness).

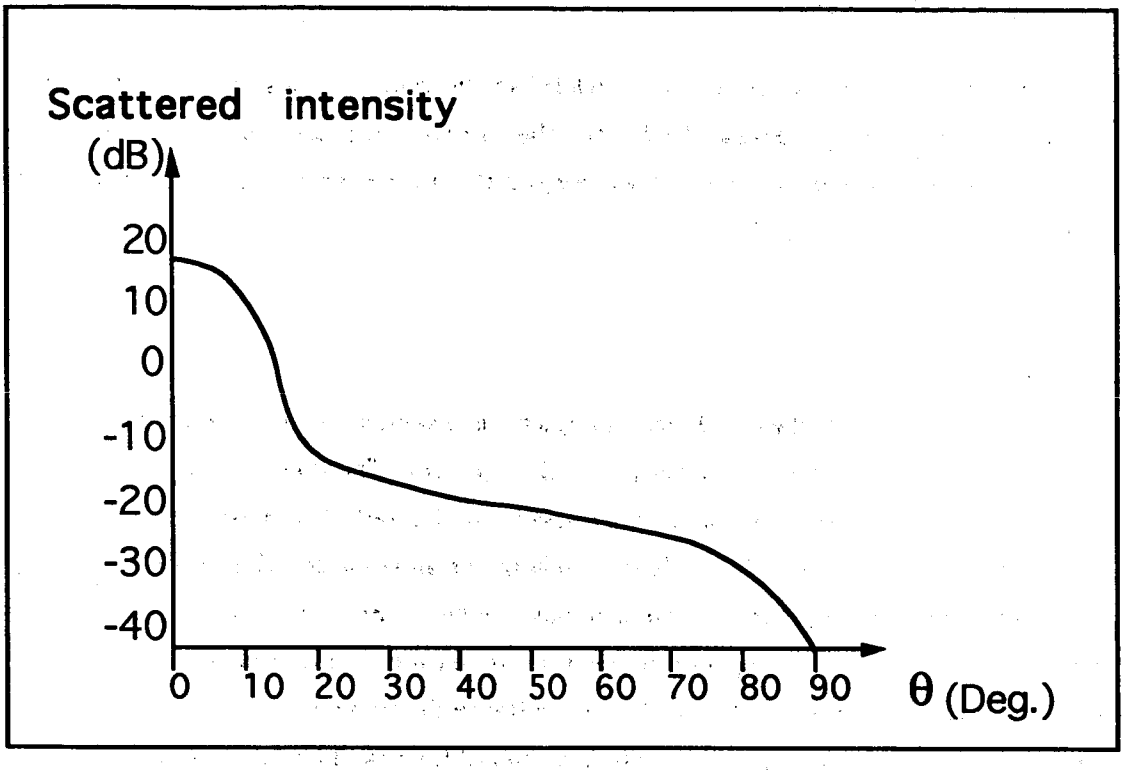


Figure 2. Typical backscattering for a composite surface model.

Emission models

Emission models are the governing models for passive sensors such as infrared sensors and radiometers (passive microwave sensing). The spectral brightness B_f (in $W.m^{-2}.sr^{-1}.Hz^{-1}$) perceived by a thermal sensor is related to the physical temperature T of the surface and to the surface emissivity coefficient $e(\theta_r, p)$ as formulated by the following equation using Planck's radiation law:

$$B_f(\theta_r, p) = e(\theta_r, p)\, 2hf^3 c^{-2} (\exp^{hf/kT} - 1)^{-1}$$

where h, f, c, and k are, respectively, Planck's constant, the frequency, the velocity of light, and the Boltzmann constant; $e(\theta_r, p)$ is the emissivity perceived from the observation angle θ_r with respect to the surface normal; and p is the polarization of the perceived radiation.

To simplify an analysis of emitted radiations, the theoretical models are divided into two categories: (1) high-frequency models and (2) low-frequency models. These models are presented in the following paragraphs.

The low-frequency model allows us to simplify the exponential term of Planck's radiation law when $hf/kT << 1$, which is also equivalent to $\lambda T > 0.77$ with λ in meters and T in Kelvin.

The Rayleigh-Jeans, or low-frequency approximation, of Planck's radiation law is

$$B_f(\theta_r, p) = 2k/\lambda^2 e(\theta_r, p) T$$

From this equation it appears that, in the microwave region, radiation emitted by the surface linearly depends on surface temperature. Therefore, considering the radiations emitted by the surface only, a radiometer provides a brightness temperature measurement T_b that depends on the surface parameters. T_b is defined as

$$T_b(\theta_r, p) = e(\theta_r, p) T$$

In the case of thermal equilibrium, emissivity is defined as [6]

$$e(\theta_r, p) = 1 - r(\theta_r, p)$$

where $r(\theta_r, p)$ is the reflectivity of the surface when illuminated with an incident angle θ_r with respect to the surface normal.

At high frequencies, Planck's radiation law is reduced to a simpler model when $hf/kT >> 1$, which is also equivalent to $\lambda T < 0.77$ with λ in meters and T in Kelvin. Using a high-frequency approximation, Planck's law is reduced to

$$B_f(\theta_r, p) = e(\theta_r, p) 2hf^3 c^{-2} \exp^{-hf/kT}$$

At high frequencies, the emissivity is adequately modelized by a Lambertian law for any surface type. Using the previous formulation, the perceived intensity at the sensor is then

$$E = (1 - \sigma_0/4)\,2hf^3c^{-2} \int exp^{-hf/kT}\,df$$

The emitted intensity is therefore a function of physical temperature and dielectric constant of the surface (through σ_0) but is independent on the observation angle.

An example of temperature brightness given by a thermal sensor is shown as a function of the surface temperature in the following figure (fig. 3).

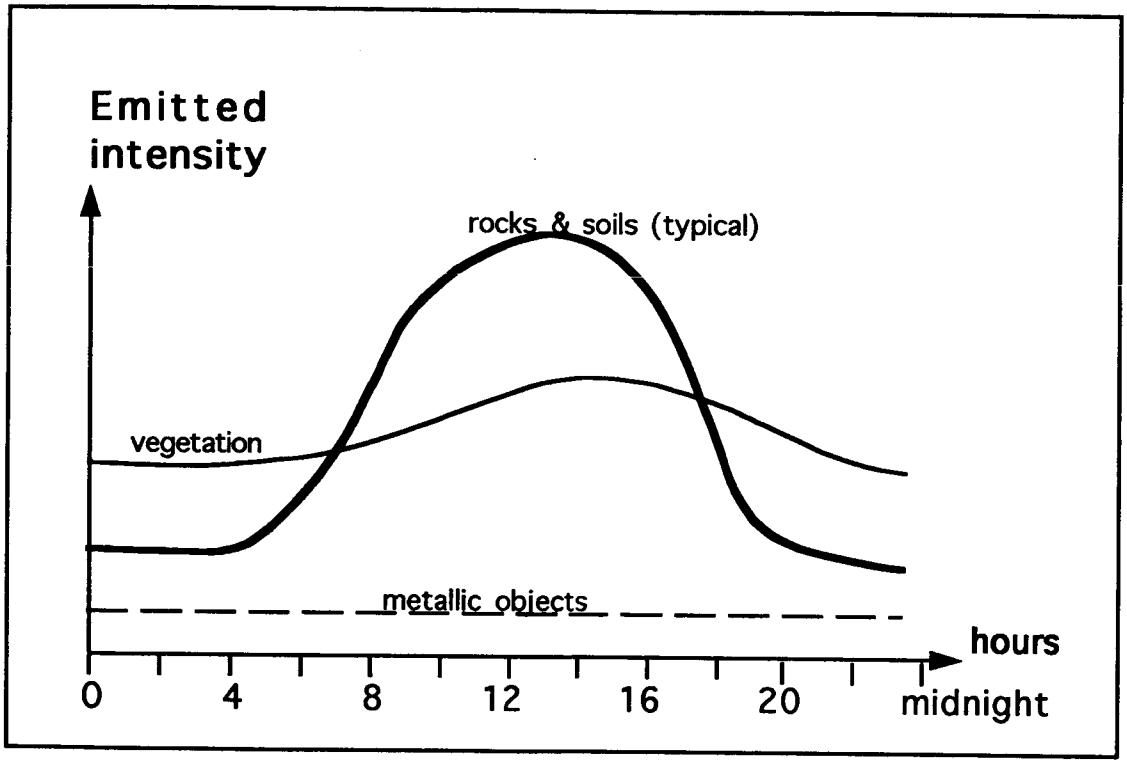


Figure 3. Radiant temperature for typical materials.

Key parameters

From theoretical modeling of scattering and emission, it appears that sensor responses are mostly dependent on surface parameters—roughness and dielectric constant—on viewing parameters —incidence and observation angles—and on sensor characteristics—frequency and polarization. The purpose of this section is to understand to what extent these parameters are affecting sensor response. This parametric analysis should then lead to the selection of a sensing configuration (sensor, sensor mode, and frequency range) that is the most sensitive to a required parameter.

By presenting experimental data as a function of incidence angle, the surface parameters, the sensor frequency, and the polarization effects are illustrated in the following paragraphs.

Roughness

The most affecting parameter is the surface roughness parameter. Roughness affects both the intensity and shape of the reflection and emission pattern. The higher the roughness, the more diffuse the scattering. Therefore, if there is a way to distinguish between a specular return and a diffuse return (which will be discussed later), surface roughness can be estimated. The perception of a surface as either smooth or rough then allows us to select the appropriate scattering and emission model with which to recover other surface parameters.

From theoretical analysis and experimental observations, the effects of surface roughness on sensor responses can be summarized as follows:

- A planar surface is smooth when
  - $k\sigma < 0.2$, where σ is the standard deviation of surface heights.
  - with active sensors, the cross-polarized return is negligible compared to direct polarization for near normal observation angles; at grazing angles, they have the same order of magnitude.
  - with active sensors, the direct polarization may be either very high or very low depending on the observation and incidence angles.
  - with passive sensors, the relative difference between perpendicular returns is high at high observation angles and very low at near normal angles.

- A planar surface is rough when
  - $k\sigma > 1.0$
  - with active sensors, the cross- and direct-polarized returns present approximately the same intensity level for any observation angle.
  - with passive sensors, the relative difference between perpendicular returns is low for all observation angles.

- An intermediate roughness surface presents intermediate behaviors for emitted and scattered intensities when
  - the surface behaves like a smooth surface at near normal angles and like a rough surface at higher angles.

- the surface presents two roughness scales. The small-scale roughness is predominant at high angles, and the large-scale roughness (or locally smooth surface of the Kirchoff theory) predominates near normal angles (< 30 deg).

Dielectric constant

The dielectric constant is a clue parameter for scene interpretation since it allows us to distinguish objects on the basis of their surface material and composition.

The dielectric constant is the second most influential surface parameter after surface roughness. This constant affects sensor response through the Fresnel reflection coefficient R. Since the Fresnel coefficient influences both reflection and emission, the dielectric constant will affect both passive and active sensing devices.

As seen in the scattering models, the intensity of reflected radiations for a planar surface is a product of a roughness term (which depends on viewing and incident angles) and a reflection coefficient. Therefore, for a given roughness and observation angle, the increase of dielectric constant will increase the sensor return.

Because the dielectric constant of material is a measure of its permittivity to incident radiations, the dielectric constant is a function of incident frequency. At low frequencies (microwaves are typical of low frequencies), the dielectric constant is highly related to the water content or moisture of materials. It provides, therefore, a useful clue for object or surface identification and is also useful when assuring the safety of mobile rovers in wet areas. In low-frequency domains, the dielectric constant of material varies from about 2 for dry soil to about 84 for water.

At high frequencies, the dielectric constant is related to material density. For most soils and materials, the dielectric constant ranges from 2 to 8 at high frequencies. In high-frequency domains, the dielectric constant is almost frequency-independent and much less sensitive to moisture.

To summarize, the effects of the dielectric constant on reflection and emission are as follows:

- For all frequencies, an increase of dielectric constant increases reflected intensities while it decreases the emitted intensities.

- The effect of dielectric constant is more sensitive at microwave frequencies than it is at visible frequencies because of its wider variation at low frequencies.

- The effect of dielectric constant is different for horizontally or vertically polarized radiations. For smooth surfaces, the reflection coefficient is higher for horizontal polarization than it is for vertical polarization. This effect is inverted, however, for emitted radiation because of the complementary behavior of emissivity and reflectivity.

Frequency

Because of the interrelationship between parameters, frequency effects present some redundancies with the effects discussed previously—and especially with the roughness effect. According to the Rayleigh criterion, as frequency increases, the surface appears rougher. In the event of a specular return, the sensor response should decrease as the surface appears rougher. In the event of a diffuse return, the sensor response is more likely to increase because of an increase of the diffuse reflection component.

A side effect of the frequency variation relates to the dielectric constant, since the dielectric constant is frequency-dependent at microwave frequencies. A frequency increase will generally produce a decrease in dielectric constant. The resulting sensor response will behave according to the dielectric effects discussed earlier. However, the dielectric constant is influencing the sensor response to a lesser extent compared when to the influence felt by the roughness effect.

Frequency effects can be summarized as follows:

- A frequency increase generally increases the diffuse reflection component and decreases the specular component due to the apparent increase in surface roughness.

- A frequency increase will also correspond to a significantly lower decrease in reflected radiations due to the dielectric constant decrease.

- High frequencies are sensitive to small-scale roughness, and low frequencies are sensitive to large-scale roughness.

- Low frequencies are more sensitive to dielectric constant and moisture variations than are high frequencies.

Polarization

Surface properties affect the polarization state of an incident wave, whether the wave was initially polarized or not. The Sun illumination is not polarized, but waves emitted by active sensors may have a controlled polarization state and might be used for surface analysis. From the sensing side, both passive and active sensors can detect the specific polarization state of received radiation. The polarization or depolarization analysis of reflected and emitted radiations is based on either active or passive sensors. This analysis can provide useful information about surface parameters.

An unpolarized or polarized illumination is reflected off a surface with a polarization or depolarization state that depends on the surface roughness scale. This, therefore, can provide a way of distinguishing diffuse and specular reflection components.

As might be expected from the Fresnel reflection coefficients, upon specular reflection, the horizontally reflected component is significantly larger than the vertically reflected component for an initially unpolarized radiation. These effects are reversed for emitted radiations, where the vertical polarization is higher than the horizontal polarization for smooth surfaces.

The behavior of initially polarized incident radiations (in the case of active sensing only) differs. Depolarization is very low for a smooth to slightly smooth surface. At near normal incident angles, however, the direct horizontal polarization HH (horizontal incident polarization and horizontal received polarization) is similar to the direct vertical polarization VV. This behavior changes at higher incident angles, where the HH polarization is higher than the VV polarization. For all incidence angles, however, the cross-polarization HV or VH (horizontally emitted polarization and vertically received polarization, and vice versa) is still much less significant than is the direct polarization.

For rough surfaces (vegetated surfaces, for example), there is little difference between polarizations because depolarization is high for any incident polarization state. Therefore, both direct and cross polarizations have similar returns and are almost angle-independent. However, the VV return is slightly higher than the HH return. The return magnitudes are also lower than those of specular reflection. Emitted radiations follow the same rule—they are unpolarized in the event of rough surfaces.

In the event of active polarized sensing, direct- and cross-polarized returns present cases of particularly interest. For smooth surfaces, an incident linearly polarized illumination is not depolarized at high incidence angles. A different effect is observed for rough surfaces, where linearly polarized illuminations are depolarized upon reflection. Thus, depolarization is related to surface roughness.

The effects of polarization are summarized as follows:

- The specular reflection of an unpolarized illumination is horizontally polarized (except for near nadir angles where there is no significant depolarization). Horizontal polarization is higher than vertical polarization (the effect is reversed for emitted radiations).

- For a smooth surface, an incident polarized radiation will be poorly depolarized, whatever its initial polarization, so that the incidence of cross-polarization is very low compared to that of direct polarization.

- For smooth surfaces, horizontal direct polarization is similar to vertical direct polarization at near normal angles but is higher at high angles ($>$ 30 deg from normal).

- The depolarization of incident linear polarized radiations becomes higher as roughness increases.

- For diffuse reflections, the incident wave—whether polarization or not—is highly depolarized so that direct- and cross-polarized returns have similar magnitudes. The same holds true for emitted radiations from rough surfaces.

- For diffuse reflections, the scattering pattern is almost angle-independent.

- For diffuse reflections, the VV polarization is slightly higher than the HH polarization.

- The relative difference between direct horizontal and vertical polarizations is related to surface roughness and the dielectric constant of the surface for both passive and active sensors.

Adaptive multisensing strategy

The envisaged approach for the perception system design, which is based on mission requirements and environmental conditions analysis, is to develop an adaptive multisensing strategy that will be determined according to illumination conditions. We focused our investigations and analyses on the following issues: What sensors and corresponding sensing modalities will lead to the best estimation of the needed surface parameters for any environmental conditions? And, how do we use the theoretical models to get information about surface parameters?

Our attempt to solve these problems led to a two-step approach. (1) Select the best appropriated set of sensors with respect to illumination conditions, sensor capabilities and complementations, and needed and known parameters. And, (2) fuse the received data to get the needed parameters. The following paragraphs describe the concepts and methods being developed for these two steps.

The selection of a multisensing configuration can be simply simulated by a table where the possible multisensing strategies are stored. Selection table columns contain a list of surface parameters—roughness, dielectric constant, range, orientation, and temperature. The table rows contain the different available sensors—active and passive microwave sensors, infrared sensors, visible sensors, a laser range finder, and laser radar. For each needed parameter, multiple sensing strategies—i.e., multiple subsets of sensor configurations—are possible. The final configuration is selected according to environmental conditions. For example, a microwave strategy would preferably be selected during a lunar night since the low frequency of emitted radiations would not be perceived using infrared sensors because of the lower temperatures of the lunar surface during the lunar night.

Once the sensing configuration is selected, the perceived data have to be fused to extract the needed parameters. We are developing fuzzy logic techniques to achieve this multisensor fusion. Compared to classical fusion techniques, fuzzy logic fusion has the following advantages [8]:

- Fuzzy logic is well suited for complementary and dependent data fusion by means of fuzzy combination rules.

- Since remote sensing models are approximate modelings of the electromagnetic scattering phenomena, surface properties cannot be determined with a high degree of accuracy. Fuzzy logic techniques allow us to process uncertain, incomplete, and ambiguous [9] measurements using simple implementation methods.

- The fuzzy description of a planetary surface is convenient for rover navigation applications since the rover does not need a highly precise description of surfaces for navigational purposes.

- Fuzzy logic also allows new information to be deduced from sensed data.

Conclusion

The objective of the conceptual perception system, which has been described, is to overcome difficulties related to the space environment illumination. A vision system should be able to perceive the planetary environment for any location and any time on the surface and to provide a description of the scene in terms of surface roughness, material identification, and surface orientation. Multifrequency and multimode sensing devices are used to achieve this analysis. The capabilities of sensors —ranging from visible to infrared and microwaves—are exploited because of complementary capabilities in terms of environmental operativeness (e.g., dust, rain, fog, night, etc.) and in terms of their sensitivity to the required surface parameters (roughness, dielectric constant, and orientation).

So far, perception problems related to space environmental conditions have been identified, an approach for overcoming these problems has been analyzed and selected, the theoretical basis for approach implementation has been settled, surface parameters and their relative influence on sensor returns have been identified and modelized for each sensor, and sensing strategies for surface parameter perception have been identified. The next steps that will lead to the development of an assured vision system are to implement and test the rules for sensor selection and sensor fusion—rules that will lead to the recovering of surface parameters.

Acknowledgment

This work has been supported by a NATO postdoctoral grant and by a NASA contract (NAS9-18440)with Universities Space Research Association (USRA).

References and bibliography

[1] K. Krishen, *Cross-polarization measurements and their relation to target surface properties.* IEEE Trans. On Antennas and Prop., vol. AP-14, no. 5, September 1966.

[2] W. H. Peake and T. L. Oliver, *The response of terrestrial surfaces at microwaves frequencies.* Report AFAL-TR-70-301, Air Force Avionics Laboratory at Wright-Patterson Air Force Base, Ohio, May 1971.

[3] F. F. Sabbins, Jr., *Remote sensing: principles and interpretation.* W. H. Freeman Ed., San Francisco.

[4] B. Siegal and A. R. Gillespie, *Remote sensing in geology.* Wiley Ed., New York, 1980.

[5] G. T. Ruck, D. E. Barrick, W. Stuart, and C. Krichbaum, *Radar cross section handbook,* vols. 1 and 2. Plenum Press, New York, 1970.

[6] F. T. Ulaby, R. K. Moore, and A. Fung, *Microwave remote sensing,* vols. 1, 2, and 3. Artech House, Norwood, Massachusetts, 1982.

[7] P. Beckman and A. Spizzichino, *The scattering of electromagnetic waves from rough surfaces.* Pergamon Press, 1963.

[8] M.-F. Collin, K. Krishen, and L.-H. Pampagnin, *Adaptive multisensor fusion for planetary exploration rovers.* International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS), Toulouse, France, 1992.

[9] T. L. Huntsberger and S. N. Jayaramamurthy, *A framework for multi-sensor fusion in the presence of uncertainty.* Proceedings of the spatial reasoning and multi-sensor fusion workshop, pp. 345-350, 1987.

Session R3: ADVANCED TELEOPERATION

Session Chair: Joe Herndon

Man-Machine Cooperation in Advanced Teleoperation

Paolo Fiorini, Hari Das and Sukhan Lee

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Abstract – Teleoperation experiments at JPL have shown that advanced features in a telerobotic system are a necessary condition for good results, but that they are not sufficient to assure consistent good performance by the operators. Two or three operators are normally used during training and experiments to maintain the desired performance. An alternative to this multi-operator control station is a man-machine interface embedding computer programs that can perform some of the operators functions.

In this paper we present our first experiments with these concepts, in which we focused on the areas of real time task monitoring and interactive path planning. In the first case, when performing a known task, the operator has an automatic aid for setting control parameters and camera views. In the second case, an interactive path planner will rank different path alternatives so that the operator will make the correct control decision. The monitoring function has been implemented with a neural network doing the real-time task segmentation. The interactive path planner has been implemented for redundant manipulators to specify arm configurations across the desired path and satisfy geometric, task and performance constraints.

I Introduction

Advanced teleoperation systems are characterized by computerized features aimed at reducing the operator's effort and enhancing his concentration during tasks. Typical features are force reflecting control joysticks, mixing of video images with computer graphics and advanced control modes such as sensor or model-referenced control. These features achieve a level of transparency between operator and task that assures good awareness of the remote task status.

The teleoperation experiments carried out in the Advanced Teleoperation (ATOP) laboratory of the Jet Propulsion Laboratory (JPL) have shown that these features are necessary for good results, but not sufficient for consistent performance. The control of the many complex features of advanced teleoperators often results in excessive mental load and fatigue even during simple realistic experiments. To avoid this potential problem, two, often three, operators are used to manage the ATOP system during training and experiments. A primary operator is in charge of the manipulation and the others carry out support functions or serve as trainers. The multioperator approach is convenient when there are no constraints on the manpower available at the control station, but, when its volume is constrained, the man-machine interface should be controllable by a single person without compromising performance and task completion time. Thus the need for automatize some of these functions.

Most of the activities of the supporting operators are quite simple and yet very time consuming and could, in principle, be replaced by computer programs embedded in the man-machine interface. Some of these operations include setting the manipulator control gains, moving cameras and lights and operating the data acquisition functions. All the values in these functions are predetermined and the only free variable is the time at which those functions have to be done. Other features that should be performed under computer supervision because of their computational requirement, are task and path planning. A man-machine interface embedding these tools would cooperate with the primary operator by providing direct suggestions, presenting command alternatives and monitoring performance fluctuations.

To test the feasibility of automatic tools of this type, we have implemented two prototypes of an automatic

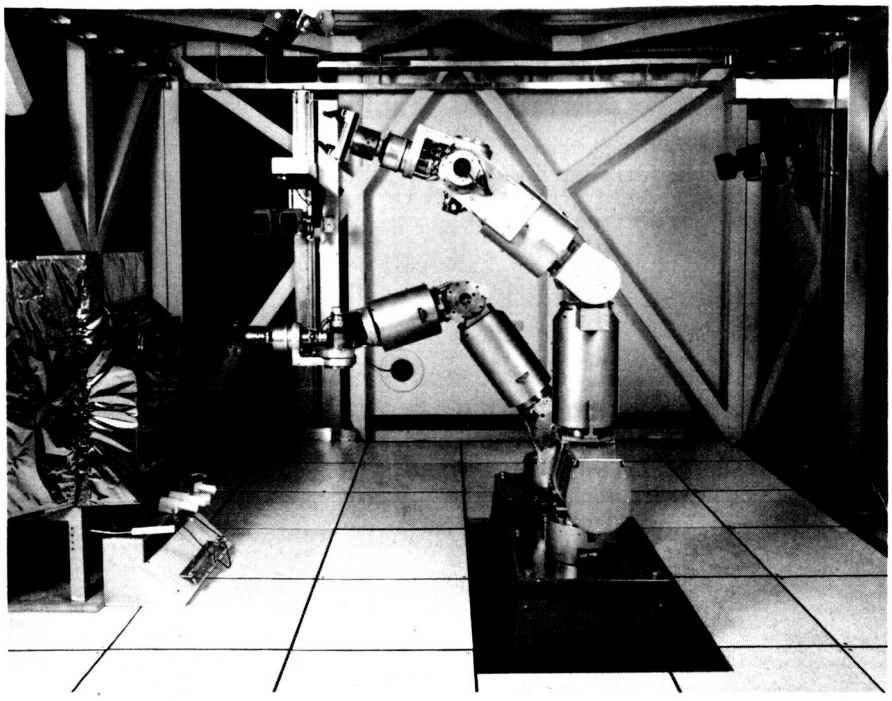


Figure 1: The JPL Dual Arm Advanced Teleoperator

monitoring program and of a redundant manipulator planner. The first tool is based on a neural-network for recognition of task phases and real time task measurement and it will permit the flexible automation of some task activities and the generation of feedback messages to the operator. In this way, a cooperative environment can be set, in which the telerobot and the operator share duties and monitoring functions of a teleoperated task. This approach would extend the paradigm of supervised control to the case in which the telerobot monitors and supervises the operator actions [4].

The second tool that we have also developed is an interactive path planner for redundant robots that allows the operator to specify arm configurations across the desired path satisfying geometric, task and performance constraints. Redundant manipulators are designed to enhance dexterity and robustness in task execution because they are equipped with more degrees of freedom (dof) than the minimum required by the task space. These manipulators allow the operator to carry out the primary manipulative task and, at the same time, to satisfy additional constraints such as avoidance of singularities and joint limits, maximization of manipulability indices, satisfaction of impedance characteristics, collision avoidance and fault tolerance. These constraints are satisfied by using the redundant dof's of the manipulator, while maintaining the position and orientation required by the task. These motions of the manipulators are called self motions and the space in which they are executed is the manipulator null space.

When redundant manipulators are used in teleoperation, there are additional technical issues that relate to the operator's role in resolving the redundancy. Information must be provided to the operator to understand the effects of self motions in A meaningful and natural form.

Next section gives a brief description of the elements of the ATOP system and of its operator interface. Our initial results in developing a task model that can be used for interface/operator interaction are presented in section III. Section IV presents a brief summary of the control problems of redundant manipulators, of the specific issues of redundant teleoperation and of our parameterization approach [7].The conclusion summarizes the paper and presents our current research and development directions.

Figure 2: The JPL ATOP Operator Interface

II The Advanced Teleoperation System

The Advanced Teleoperation Laboratory (ATOP) is physically divided into two parts: the remote site and the local site. The main features of the remote site are:

1. A *Camera Positioning System* consisting of a steel frame capable of supporting five movable TV cameras. Currently, the frame supports three cameras mounted on pan/tilt platforms attached to computer controlled beams moving on the frame sides and ceiling. A monoscopic camera is mounted on the ceiling and one on a side beam, while a stereo camera is mounted on the other side beam.

2. A *Dual Arm System* consisting of two AAI 8 degree-of-freedom redundant manipulators, two Model C JPL Smart Hand and two sets of Universal Motor Controllers (UMC) systems (fig. 1). The UMC electronics performs the joint servo, the inverse and direct kinematics of the redundant manipulators, and handles the communication with the smart hands and with the local site via fiber optic serial lines. Several advanced control features are available in the dual arm system, such as *contour following, shared compliance* and *harmonic motion generator* [1], [8].

3. *Controlled Light Sources* to create viewing conditions similar to those found in a space environment.

The local site of the ATOP system consists of the control room with the operator interface (fig. 2). This interface has evolved following the development and the integration of new technologies into the teleoperation system and is organized in three main subsystems:

1. **A Data Interface** consisting of:

   (a) *Parameter acquisition* interface to enter robot configuration and control parameters.

   (b) *Data acquisition* interface to handle the communication with the smart hand and process incoming force and torque data.

   (c) *Programming, debugging and set up* interface for the development, monitoring and setup of the control programs for the manipulators.

   (d) *Sensors* interface to visualize in real time force and torque data collected by the manipulators sensors.

2. **A Video Interface** composed of:
   (a) *Video Monitors*, displaying two monoscopic and one stereo views of the remote site,
   (b) *Computer Graphics Simulation*, equipped with predictive display, for time delay compensation and operator training.
3. **A Manual Interface** consisting of:
   (a) *Force Reflecting Hand Controllers* (FRHC), driven by a pair of UMC's to command the movements of the robots and to feed back force information to the operator.
   (b) *Camera Gantry Controls*, to move the beams holding the cameras to the desired viewing position.
   (c) *Camera Controls* to adjust pan, tilt, focus, zoom and iris of the cameras.
   (d) *Foot Pedals* for the power tools in the remote site.

III AUTOMATIC SUPERVISION

An efficient single-person teleoperation interface must perform some of the functions assigned to the operators of a multi-operator interface. To experiment with this idea, two different functions are under study: the first one is the automatic display of camera and control menus during a task, and the second is the generation of feedback messages to the operator. In the first case, the interface must automatically show the operator the appropriate command menus for the specific task's phase. In the second case, the interface must provide the operator with performance feedback messages derived from a stored model of the task execution. In both cases, a key element of these advanced tools is a program that can follow the development of a teleoperated task by segmenting the sensory data stream into appropriate phases.

A task segmentation program of this type has been implemented by means of a *Neural Network* architecture [2] and it is able to identify the segments of a *peg-in-hole* task. With this architecture, the temporal sequence of sensory data generated by the wrist sensor on the manipulators are turned into spatial patterns and a window of sensor observations is associated to the current task phase.

This problem is referred in the literature as that of *learning time sequences* and has been approached primarily with two architectures: *Time-Delay* and *Partially Recurrent* networks [5]. Partially Recurrent Networks better represent the temporal evolution of the task since they include in the input layer a set of nodes connected to the output units, to create a context memory. These units represents the task phase already executed – *the previous state*. A set of **fixed** weight connections have been established among the output and context layer units (see figure 3).

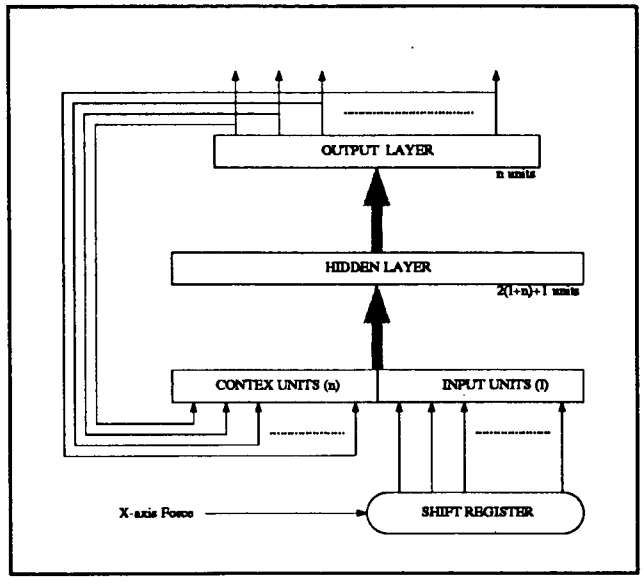


Figure 3: Partially Recurrent Neural Network

Training is carried out to associate the previous state and the window of sensor signals to the current state:

$$\sigma_1(t-\Delta t),..,\sigma_n(t-\Delta t),x(t-(l-1)\Delta t),..,x(t) \rightarrow \sigma_1(t),..,\sigma_n(t)$$

where $(\sigma_1(t),\ldots,\sigma_n(t))$ is the state corresponding to the x-force sensor signal $x(t)$. The window length (l value) is a critical design factor.

The Partially Recurrent network gave good results both in training and in simulation and it has been interfaced to the ATOP telemanipulator system for real-time tests. The ATOP configuration is significantly different from the one used for the training data and therefore these tests also verified the robustness of the approach to hardware variations.

Figure 4 shows as a dotted line the output of the real-time segmentation performed by the network during an actual peg-in-hole task. The solid line represents the correspondence between task's phases and data. The time response of the network is evident in the lag between homologous transitions between the solid and the dotted lines and it depends primarily on the computer used to

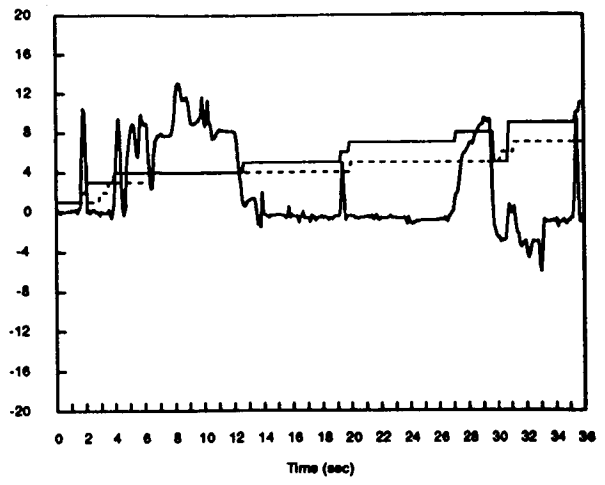


Figure 4: Segmentation in the real-time experiment

collect data and perform the network calculations. Several experiments have been carried out and the results have been quite encouraging with a percentage of correct segmentations approximately equal to 65%. Figure 4 refers to the a typical Peg-in-Hole phases' sequence [3]. Approximately at time 29 sec the network misclassifies the end of the *extract* phase (index n. 8) that should have occurred at time 34 sec. The network showed also unexpected capabilities to recover after misclassification and it was also able to follow tasks whose phases' sequence varied from the training one. During peg extraction, for example, if the operator decided to regrasp the peg, the network was sometimes able to make the transition from *extraction* to *insertion* and again to *extraction*. These tests were quite encouraging and this type of architecture can provide a building block for the automatic tools of an advanced teleoperation interface.

IV Cooperative Redundancy Management

For redundant manipulators it is not possible to formulate a closed form solution of the inverse kinematic problem and many control algorithms use a mixture of velocity based control and optimization procedures. The inverse Jacobian is used to compute the velocity transformations and a local optimization of some evaluation criteria is used to determine the values of the redundant joints in the arm null space [9], [6]. The above velocity-based kinematic control has several drawbacks, including computational cost, numerical instability when the Jacobian inversion is performed near singularities and the lack of global optimization of the arm configurations.

To eliminate these problems, a parameterization approach has been proposed for the control of redundant manipulators [7]. The redundant joints are considered as parameters of a non-redundant manipulator for which we can obtain a closed form solution of the inverse kinematics. This approach transforms the problem of controlling a redundant arm into that of selecting optimal values for the configuration parameters, i.e. the redundant joints, along the required trajectory.

This method has computational and numerical advantages over velocity-based kinematic controls and it provides the necessary support for implementing cooperative redundancy management in teleoperation. In fact, the operator can be shown a display of the parameter space in which a surface representing a performance criterion is visualized. Every point on the surface is associated with values of the redundant joints and the operator can immediately determine the best configuration. The values of the redundant joints are the parameters that are used to compute the inverse kinematics of the associated non redundant manipulator.

More specifically, a null space manifold is formed in the parameter space with the consideration of individual joint limits, and characterized by an artificial potential field representing the performance associated with the individual null space joint configurations. A null space manifold can be easily formed by varying the parameter values within their limits, and by checking the availability of a solution through the position-based kinematic solution. The manifold can be incrementally updated and animated along the given task trajectories. The artificial potential function is then formed over the null space manifold based on a combination of several desired attributes such as proximity to joint limits, proximity to singularities, and measures of kinematic and dynamic manipulability. The potential function represents the performance criterion for the selection of the joint configuration. It can be used either automatically by the system, following a local gradient search, or manually by the operator. In this case, globally optimal joint configurations may be selected by extrapolating the variations of performance associated with the selected parameter values, or by analyzing the variation of the potential function at successive task points along the given task trajectory. In short, the parameterization method allows the visualization of manipulator internal performance through the display of potential functions in the parameter space. This provides a medium for an interactive and cooperative interface for redundant telemanipulator planning, through which the

operator can decide whether, when and how to reconfigure the arm for optimal task execution.

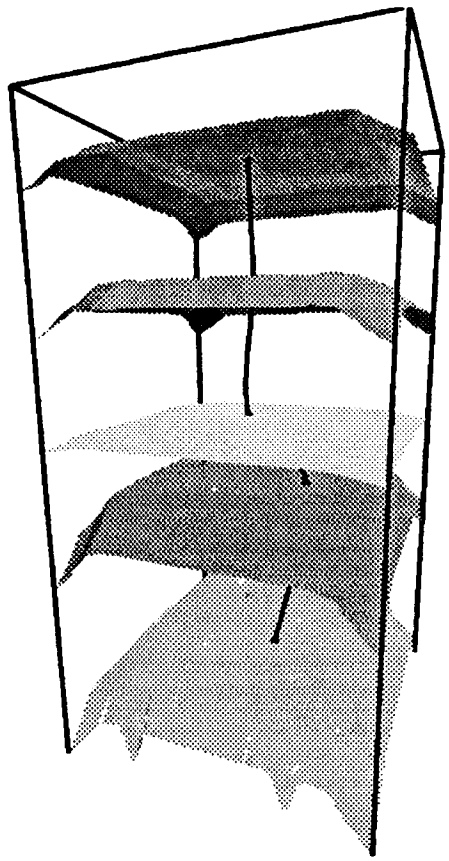


Figure 5: Criterion functions on the 2-D null space

The AAI arms used in the ATOP laboratory are decomposable arms and particularly suited for the proposed parameterization method that can be applied to each one of the two subarms in which the manipulators can be partitioned. In this case, the parameterization method is very simple because a closed form of the parameterized inverse kinematic solution can be readily obtained from each subarm and the null space manifold in the parameter space can be easily described.

The AAI arm is an 8 dof manipulator, where the 4 lower revolute joints are configured as a cascaded spherical arm for positioning and the 4 upper revolute joints are configured as a Z-X-Y-Z wrist for orientation. The AAI arm has no link offsets and since the 4 wrist axes intersect at one point, the arm is decomposable. This manipulator permits an easy kinematic control because of its zero offsets and orthogonal mechanical structure and it also achieves a high dexterity and singularity avoidance.

The proposed parameterization method has been successfully applied to the derivation of the closed form, parameterized inverse kinematic solution of AAI manipulators. The derivation was simplified by taking advantage of this arm decomposability: the inverse position transformation uses two joints as parameters, one in the lower arm (first 4 joints) and the other in the upper arm (last 4 joints). A potential function over the solution manifold is then mapped onto the parameter space for a given task point, as illustrated in fig. 5.

In our current implementation, the operator is presented with a 3-D graphic display of the criterion functions plotted over the null space of the robot for five different end effector positions and orientations. Corresponding to each end effector position and orientation, a surface is plotted showing the value of the criterion function over the two dimensional null space. The operator can specify the weights for each criterion function for the different surfaces. The display seen by the operator is shown on Fig. 5.

The input to the program is with a graphic interface and it consists values of the redundant joints with which to start or to tune the computation of the optimal trajectory. This approach to redundant path planner offers a number of advantages to the operator for constructing feasible paths during a task. In particular, the operator has complete control in selecting the robot configuration and the process is carried out in real-time since the tool provides immediate visual feedback on the quality of the selected parameters.

V Conclusions

In this paper we have described our current efforts to improve the communication between the operator of a telerobotic system and the system itself. The long term objective is to establish a duality between the human supervision of a telerobot, as in the supervisory control paradigm, and the automatic support to the user of an advanced teleoperator. Two main areas of research have been described. The first one aims at developing techniques for supporting the operator during task execution. The type of support envisioned within the interface will consist of task segment identification for the automatic setting of system parameters and for performance monitoring. The second research effort is concerned with the development of an interactive path planner that cooperates with the operator in the selection of the best path and configurations of a redundat telemanipulator. These automatic features will be an asset in all phases of tele-

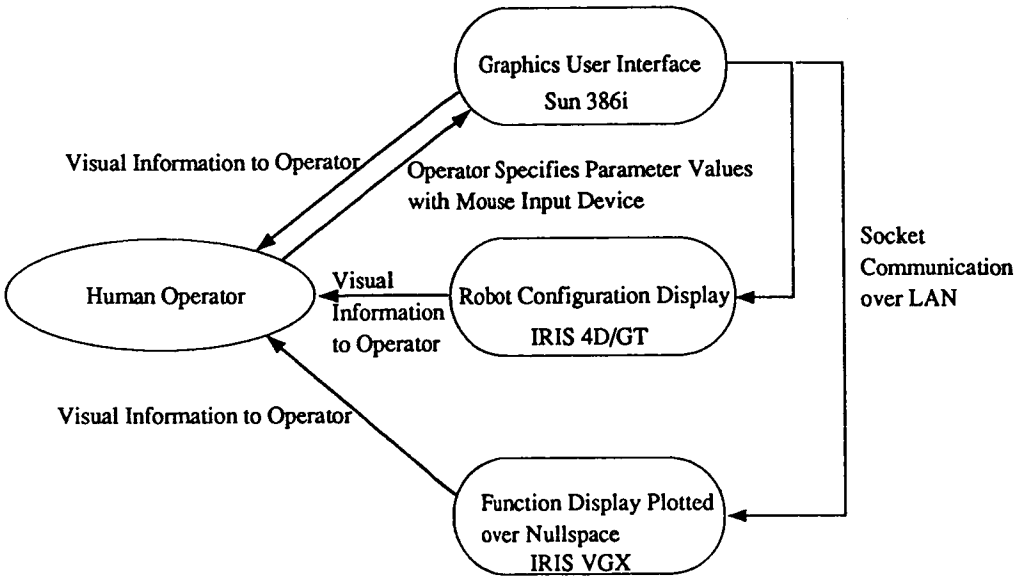


Figure 6: Operator Interaction with the Planner

operation and will be essential in supporting real teleoperated tasks.

VI ACKNOWLEDGMENT

The authors would like to thank Antonio Giancaspro for his contributions to the neural network models. The research described in this paper has been carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautic and Space Administration.

REFERENCES

[1] A.K. Bejczy, Z. Szakaly, and W.K. Kim. A laboratory breadboard system for dual arm teleoperation. In *Third Annual Workshop on Space Operations, Automation and Robotics*, JSC, Huston, TX, July 25-27 1989. NASA Conf. Publication number 3059.

[2] P. Fiorini et al. Neural network for off-line segmentation of teleoperation tasks. In *IEEE International Symposium on Intelligent Control (ISIC-92)*, Glasgow, Scotland, August 1992.

[3] B. Hannaford, L. Wood, D. McAffee, and H. Zak. Performance evaluation of a six-axis generalized force-reflecting teleoperator. *IEEE Transaction on Systems, Man and Cybernetics*, 21(3), May/June 1991.

[4] S. Hayati and S.T. Venkataraman. Design and implementation of a robot control system with traded and shared control capability. In *IEEE International Conference on Robotics and Automation*, volume 3, Scottsdale, AZ, May 14-19 1989.

[5] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA, 1991.

[6] C. A. Klein and C-H. Huang. Review of pseudo-inverse control for use with kinematically redundant manipulators. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-13(3), March 1983.

[7] S. Lee and A.K. Bejczy. Kinematic solution of 8 degree-of-freedom AAI arm. In *IEEE International Conference on Robotics and Automation*, Sacremento, CA, April 1991.

[8] Z. Szakaly, W.K. Kim, and A.K. Bejczy. Force-reflective teleoperated system with shared and compliant control capabilites. In *NASA Conference on Space Telerobotics*, volume 4, Jet Propulsion Laboratory, Pasadena, CA, January 31 1989.

[9] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. on Man-Machine Systems*, MMS-10(2), June 1969.

Integration of Advanced Teleoperation Technologies for Control of Space Robots

Michael J. Stagnaro
NASA / Johnson Space Center
Automation & Robotics Division
(713) 483-1520
Email: stagnaro@ctsd2.jsc.nasa.gov
August 5, 1992

ABSTRACT

Teleoperated robots require one or more humans to control actuators, mechanisms, and other robot equipment given feedback from onboard sensors. To accomplish this task, the human or humans require some form of control station. Desirable features of such a control station include operation by a single human, comfort, and natural human interfaces (visual, audio, motion, tactile, etc.). These interfaces should work to maximize performance of the human/robot system by streamlining the link between human brain and robot equipment.

This paper describes development of a control station testbed with the characteristics described above. Initially, this testbed will be used to control two teleoperated robots . Features of the robots include anthropomorphic mechanisms, slaving to the testbed, and delivery of sensory feedback to the testbed. The testbed will make use of technologies such as helmet mounted displays, voice recognition, and exoskeleton masters. It will allow for integration and testing of emerging telepresence technologies along with techniques for coping with control link time delays.

Systems developed from this testbed could be applied to ground control of space based robots. During man-tended operations, the Space Station Freedom may benefit from ground control of IVA or EVA robots with science or maintenance tasks. Planetary exploration may also find advanced teleoperation systems to be very useful.

1.0 INTRODUCTION

Remotely controlled robots may be successfully applied in hazardous environments such as high radiation zones, deep sea locations, earth orbit, and extra-terrestrial sites. Three dominant control modes are teleoperation, supervised autonomy, and shared control[1]. Teleoperation is characterized by direct human-in-the-loop manual control and small time delays (< 1 sec.). In supervised autonomy, commands are generated by the operator and sent to the robot control system for execution. Shared control makes use of both teleoperation inputs and an autonomous robot control system. In each of these modes, a human operator is involved and must interact with some form of computer based control station.

Figure 1.1 illustrates a spectrum of technologies which may be used in a remote robot control station. At one end are conventional technologies such as hand controllers, 2-D video, keyboards, and computer monitors. The other end is labeled as telepresence technologies and includes force reflective exoskeletons, stereo (3-D) video, voice recognition, and synthetic speech.

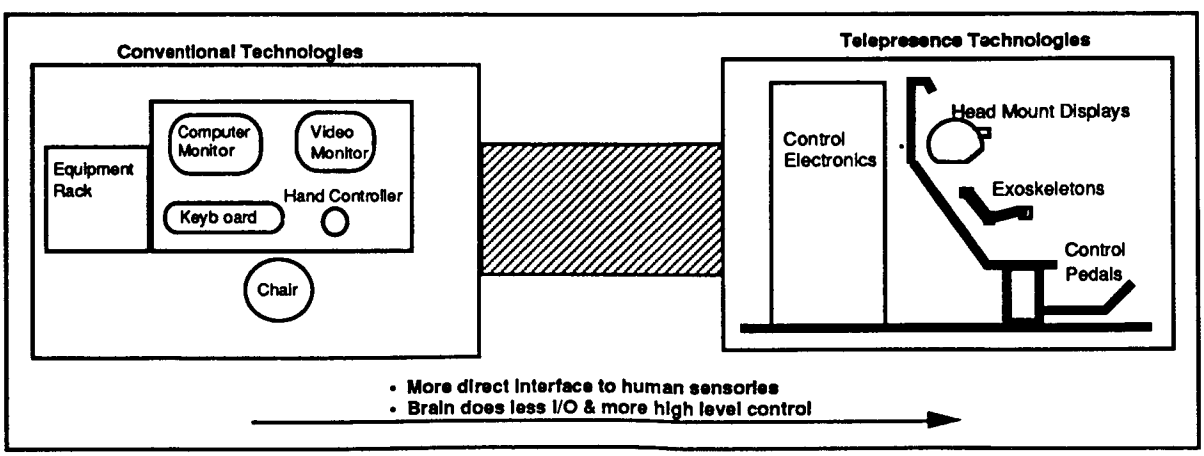


Figure 1.1 - Remote Robot Control Technology Spectrum

Telepresence can be defined as the sense of being physically present with object(s) at a remote site[3]. Telepresence technologies attempt to immerse a human operator in the remote environment with actuator control and sensory feedback devices which closely interface to the human central nervous system. Robots at the remote site are designed with anthropomorphic actuators and sensory devices, such as stereo camera pairs and tactile sensors.

Telepresence technologies offer interfaces to the human sensory system which are more direct than those of the conventional technologies defined. This frees the brain from many unnatural input/output conversion tasks, allowing more concentration on higher level control and process oriented tasks. The result is a more intuitive way of controlling remote robots.

Major challenges facing telepresence technologies include working with time delays, increasing video resolution and field of view, and provision of adequate force/torque and haptic feedback.

2.0 PROJECT OVERVIEW

2.1 Objectives

The primary project objective is to develop a teleoperator control station testbed which makes use of telepresence technologies. This testbed is referred to as the Teleoperated Robot Interface Platform (TRIP) and should provide teleoperation capabilities for two JSC development robots. One robot is the Dexterous Anthropomorphic Robotic Testbed (DART)[4]. DART is a dual-arm, dual-hand robot with a camera platform which provides stereo video. It will be able to operate under human control augmented by on board intelligence for use in development of IVA and EVA robotic systems. The second, AERCAM, is a prototype mobile camera platform capable of teleoperation. Initially, this prototype will be flown on an air bearing floor at JSC.

In a more general sense, TRIP will be used as a testbed for emerging telepresence technologies, such as head mounted displays, exoskeletons, and programming systems. In addition, TRIP will allow testing of proposed solutions to the problems induced by control link time delays. Systems developed with the TRIP testbed should support future operations on board the Space Station Freedom, including the possibility of ground control using shared control techniques.

2.2 Goals & Constraints

TRIP development goals include flexibility, ease of use, and growth paths. A flexible system will support the testbed objectives through maximum use of standard hardware and software interfaces, a modular approach to system design, and the use of software for most calibration tasks. The system should also allow for ease of development and use by minimizing and simplifying the software learning curves. Ease of use will support tight schedules and minimal manpower. A system designed with growth paths will foster an evolutionary development by protecting both hardware and software investments. Design for growth seeks to avoid obsolescence by choosing established software tools with supported growth paths and by avoiding hardware with closed or unsupported architectures.

Development constraints consist of cost and system performance. Design must be sensitive to costs by maximizing system capability given current year funding. Basic system level results should not depend on large amounts of future funding. Basic performance requirements, such as data rates and connectivity, must also be satisfied. Optimization of performance variables at the expense of system flexibility will be avoided unless required.

2.3 Facilities & Support

TRIP is under development in the Dexterous Robotics Lab of JSC's Automation & Robotics Division. The two target robots are also under development in division labs. To date, all primary design and development work has been conducted in-house by JSC civil service staff members. Only a limited amount of contractor support has been available or used.

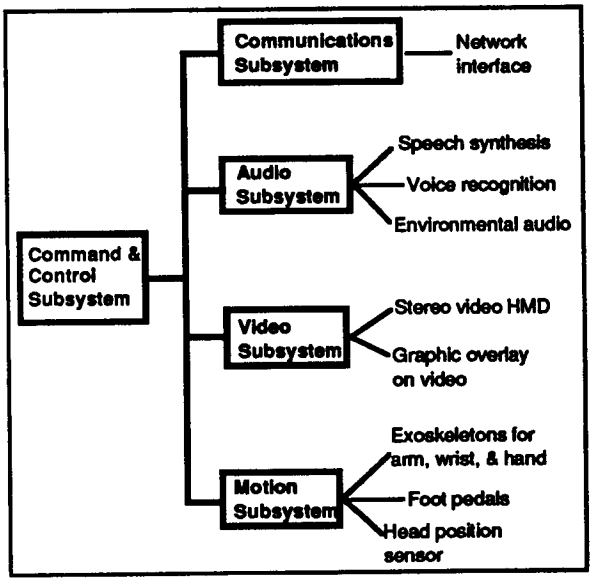


Figure 3.1.1 - TRIP Subsystems and Technologies

3.0 SYSTEM DESCRIPTION

3.1 Organization

TRIP is the integration of assorted telepresence technologies as illustrated in figure 3.1.1. These include exoskeletons for the operators hand, wrist

and arm joints, a head mounted display for viewing stereo video and graphics, speech synthesis and voice recognition systems, and a network interface for passing data to/from a remote robot. These systems should work in concert to provide intuitive control of a remote robot by one human operator. TRIP is organized into five subsystems, each consisting of hardware and associated software.

3.2 Hardware Architecture

The hardware architecture is shown in figure 3.2.1. Selections were driven primarily by the flexibility goal along with the availability and cost of both software and special purpose boards (video, audio, etc.). Hardware cost and growth path trends were also considered.

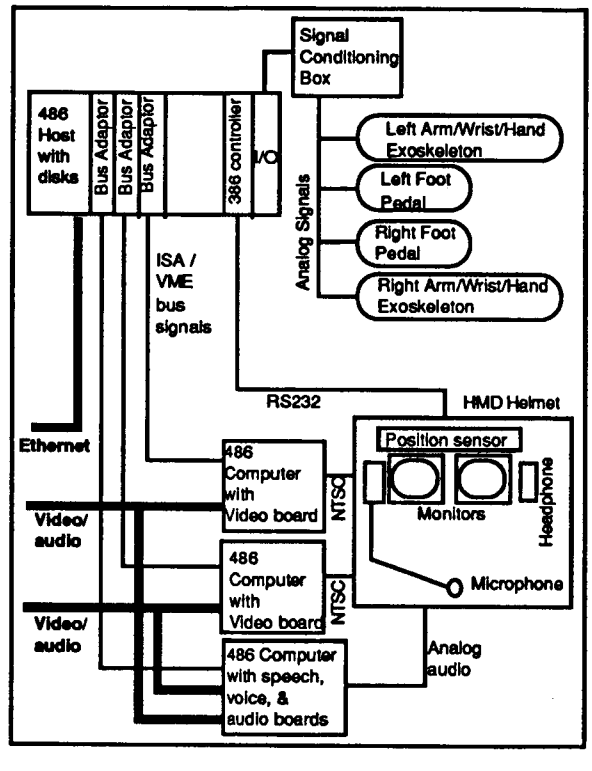


Figure 3.2.1 - Hardware Architecture

The hardware architecture consists of exoskeletons, a head mounted display, a chair platform with control pedals, i486 & i386 microprocessors in ISA & VME buses, and assorted I/O, audio, and video boards. A single family of processors was selected to minimize software learning curves and keep costs relatively low. The ISA bus offers reasonable performance and a myriad of low cost, special purpose I/O boards to choose from. The VMEbus offers an industry standard with high bandwidth performance, extreme flexibility, and a large number of I/O and processor boards. Selections of boards for both buses were based on a balance of cost, flexibility, and performance. Software driver libraries were also considered in the board selections as this represents a potentially labor intensive set of development tasks.

Exoskeletons are attached to a body harness and gloves which the operator wears. An analog signal conditioning box provides power to the exoskeletons and filters the signals produced by potentiometers and hall effect sensors. These signals are then read by an analog to digital (A/D) converter board in the VMEbus. An embedded i386 computer processes raw data from the A/D board and makes the results available on the VMEbus.

The head mounted display (HMD) consists of monitors, optics, a position sensor, headphones, and a microphone. The position sensor reports roll, pitch, and yaw to an embedded i386 computer via an RS-232 link from its own processor based control box. Two i486 based computers deliver video with text or graphics overlay to the monitors. The headphones are driven by a third i486 computer which handles speech synthesis and other audio feedback functions. This computer also handles voice recognition tasks, making use of the microphone. All three computers use ISA to VME bus adaptors to communicate with the VMEbus. Video and audio signals from a remote robot are currently transmitted through dedicated channels.

A chair platform includes control pedals and a transmitter for the HMD position sensor. Potentiometers in the pedals are powered and read by the same hardware used with the exoskeletons. Signals are also processed by the embedded i386 computer and results are available on the VMEbus. The HMD position sensor transmitter has its own power supply and processor based control box.

An i486 embedded in the VMEbus serves as the command and control computer of TRIP. Through the VMEbus, this computer can communicate with all subsystems and coordinate their interactions. In addition, this computer handles all data communications with remote robots through an ethernet network board and a single dedicated cable.

3.3 Software Architecture

Figure 3.3.1 displays the current high level software architecture. Software modules are shown in round-edge rectangles, and hardware is represented in square-edge rectangles.

The software architecture consists of control and configuration tasks, command routers, command handlers, bus data exchange drivers, RS-232 interface drivers, and TCP/IP network interface drivers.

The control and configuration tasks reside on an embedded i486 computer in the VMEbus. These allow the operator to calibrate subsystems and define subsystem interaction rules. Control tasks

coordinate the interactions between various subsystems.

Command routers serve two functions. First, process data from input devices to produce specific subsystem commands. Second, route these commands to the appropriate subsystem message areas. Command routers are used with the voice recognition system and all motion input devices.

Command handlers accept commands or messages from command routers and execute the commands or answer messages. Command handlers are used with output systems such as video and graphic overlays, speech synthesis, and environmental audio.

Bus data exchange drivers allow commands and messages to be physically exchanged between the ISA buses and the VMEbus. Both embedded and external computers use these drivers

Drivers for the RS-232 ports are used by the embedded i386 computer to communicate with the HMD position sensor controller. Commands can be sent to the controller and raw orientation data is read. This raw data is also parsed and made available to bus data exchange drivers.

Network access is provided by interface drivers using the TCP/IP protocols. These are used by the communication subsystem to transmit commands to or receive messages from a remote robot. They are also used to handle commands from TRIP subsystems and route commands from the robot to appropriate handlers.

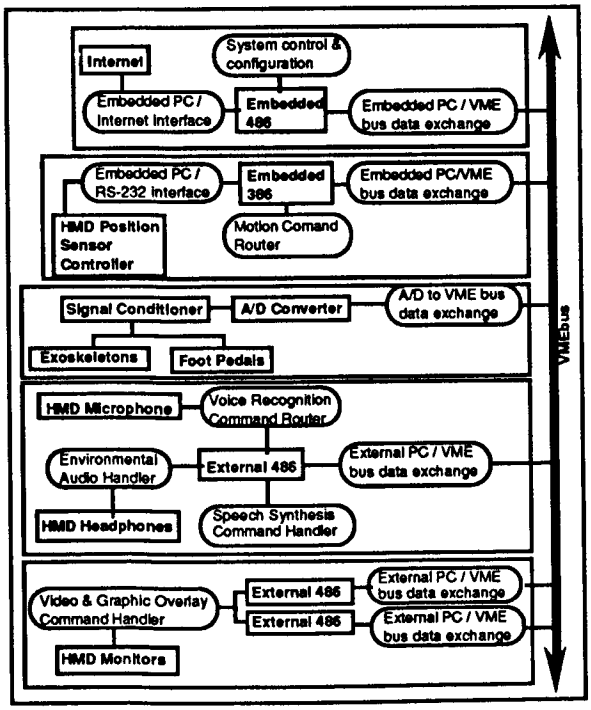


Figure 3.3.1 - Software Architecture

3.4 Software Tools

Current software tools comprise two operating systems and three compilers. Windows 3.1 and iRMX are the operating systems and they support development with Visual Basic, Borland C++, and iRMX C.

Windows 3.1 provides cooperative, event driven multitasking with an easy to use graphical human interface. A large number of board level drivers directly support this operating system and development with its standard user interface. Windows 3.1 also provides a growth path to a 32 bit preemptive multitasking/multiprocessing environment with Windows NT. Windows NT will be backward compatible with 3.1 and will use the same graphical interface standards. It should be available at the end of 1992.

iRMX provides 32 bit mode operation of the Intel microprocessors and real time task scheduling for low level, time critical tasks. A unique feature of this operating system is its ability to run Windows as a task and communicate between the two operating systems. This enables the best of two worlds: 32 bit hard real time tasking and an easy to use standard interface.

Visual Basic (VB) is an object oriented visual development environment for the Windows operating system. Objects can send or receive messages, and events can be used to trigger user developed code or operating system calls. The syntax is similar to that of Basic, but the code structure and object orientation endow it with many features found in C++. The visual development environment lends itself to very rapid prototyping of code and almost effortless user interface development. VB does not support some of the low level capabilities found in C or C++, but Dynamic Link Library (DLL) functions written in C or C++ may easily be called. Operating system functions may also be called directly from VB. Dynamic Data Exchange (DDE), a client/server intertask communication protocol, is also supported and easy to use.

Borland C++ is an object oriented C development environment which supports development for the Windows operating system. Specific to this project, it allows the development of low level DLL functions which may be called from VB code. Borland supplies an efficient code development environment with extensive debugging support. The object orientation promotes development of complete code modules which are easy to reuse and build upon.

iRMX C is a compiler and assorted tools for developing C code task modules which run in the iRMX real time operating system. These modules will accommodate time critical, low level functions as required by the TRIP. These functions may

communicate with Windows hosted code to report system status, alarms, or data needs.

3.5 Floor Layout

A planform view of TRIP hardware is displayed in figure 3.5.1. Shown are the chair platform, a 19 inch equipment rack, and two development work sites. An adjustable chair is mounted to the platform and serves as the operator work site. Pedals, exoskeletons, and the HMD are all connected from the chair platform to equipment in the 19 inch rack. The rack contains all TRIP computers along with audio and video ancillary equipment. Two development work sites each incorporate a keyboard, mouse, and two SVGA monitors. One work site supports development on VMEbus subsystems which include command and control, communication, and motion. The other work site supports development on the audio and video subsystems.

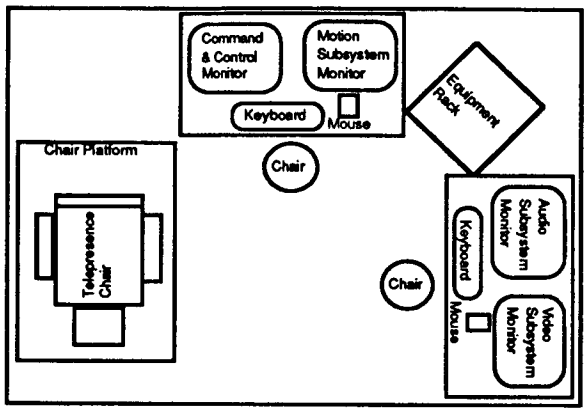


Figure 3.5.1 - Floor Layout

4.0 SUBSYSTEM DETAILS

4.1 Motion Subsystem

The motion subsystem generates commands to control position or rate of all articulated members on a remote robot. Robot members include arms, hands, torso's, camera platforms, and mobile bases. The operator controls these using a combination of exoskeletons, foot pedals, and position sensors. Force reflection and tactile feedback for the operator are planned as growth paths in the arm and hand exoskeletons of this subsystem.

Current components of this subsystem are detailed in figure 4.1.1. It consists of an embedded i386 computer, an HMD position sensor, an A/D board with a signal conditioning box, exoskeleton arm and hand masters, and foot pedals.

An embedded i386 based computer from the Radisys Corporation is used for subsystem processing and control. It runs at a clock speed of 25 MHz, contains 8 MB of DRAM, a keyboard controller, serial ports, and an ISA compatible private bus which supports a 40 MB hard disk and a super VGA controller board. It boots with a PC/AT compatible BIOS ROM and supports a number of operating systems. Radisys also supplies low level functions which allow direct access to all VMEbus memory spaces. These functions are compatible with TRIP software tools.

A Logitech 6D Mouse is used to sense the HMD position and orientation. It makes use of an ultrasonic transmitter and receiver triangles to determine location in Cartesian space (x, y, z) and orientation (roll, pitch, yaw) as Euler angles or quaternions. Also included is a dedicated control processor which communicates with the host processor via RS-232. Advantages of ultrasonics over magnetic sensors include reduced lag times and no interference from metal structures. A disadvantage is that the full range (0-360 degrees) of Euler angles is not supported, although TRIP does not require the full range for HMD tracking.

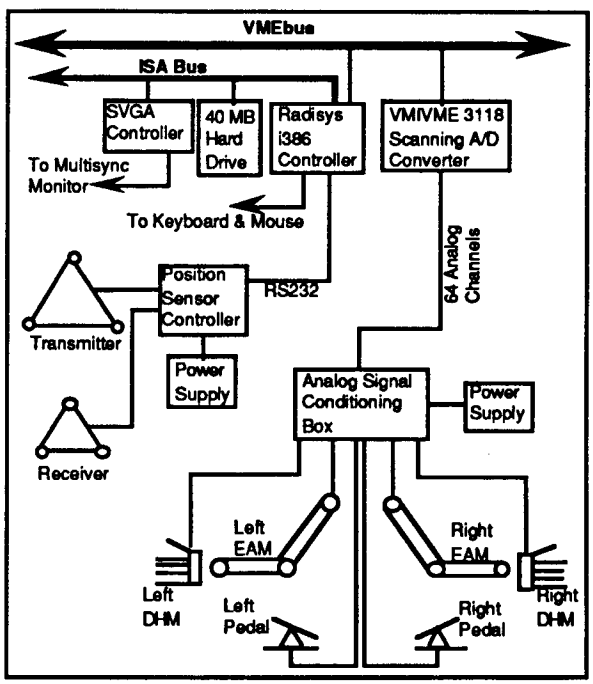


Figure 4.1.1 - Motion Subsystem Block Diagram

A VMIVME 3118 scanning A/D board is used in conjunction with a signal conditioning board developed in-house to support 64 differential channels. The A/D board interfaces to the VMEbus via control registers and a dual ported RAM data buffer. The signal conditioning board is housed in a separate box with a dedicated power supply. The board low pass filters (10 Hz) each channel and buffers the signal lines to the A/D board. It also supplies power to sensors on the exoskeletons and foot pedals.

Two exoskeleton arm masters (EAM) and two dexterous hand masters (DHM) from Exos, Inc. are

utilized in TRIP. The EAM provides precise measurements of human shoulder and elbow joint angles. Potentiometers are currently used to sense the angles. The DHM uses hall effect sensors to measure joint angles of the human hand. A GripMaster (GM) is integrated into each DHM to measure wrist motion. Development continues to be funded by NASA with future objectives including the addition of sensory feedback in all areas. The current TRIP design will be capable of integrating these planned improvements.

Foot pedals can be used to control robot torso and/or mobile base motion. In both cases, potentiometers are used to sense ankle joint angles which provide rate and directional control of robot motors. These pedals are attached to the chair platform.

Software tasks running on the i386 read and process raw data from each of the input devices. One task reads a data buffer on the A/D board and processes for joint angle or rate commands. Processing includes some simple (i.e. boxcar) noise filtering and any required coordinate transforms. The A/D board continuously scans all active channels and updates the entire data buffer at about 800 Hz. A second task on the i386 reads and parses data from an RS-232 port to determine roll, pitch, and yaw of the HMD. This port communicates with the position sensor controller which continuously updates position readings at about 50 Hz. A third task on i386 accepts processed data from the other tasks and routes it to the appropriate command handlers (i.e., communication, graphic overlay, etc.) using bus data exchange drivers.

4.2 Video Subsystem

A video subsystem handles all live video signals along with any computer generated graphics. Video is supplied by camera pairs on the remote robot which are designed to provide stereo video to the human eyes. Computer generated graphics and/or text may be calibrated to the video and overlayed to provide visual feedback, simulation results, or task tools to the operator.

Figure 4.2.1 is a diagram showing half of the video subsystem. Each half feeds one of the operators eyes, and both halves are identical. Components include a helmet with HMD's, a video scan converter, a frame grabber and video compression board, an SVGA graphics board, an i486 based ISA bus computer, and ISA to VMEbus adaptor boards.

A head mounted display helmet from Virtual Research is currently in house. It incorporates headphones, the Logitech 6-D Mouse receiver triangle, two color LCD displays (360 x 240 pixels), and wide angle optics from Leep Systems. The helmet is designed for comfort is extremely easy to don and doff. The Leep Systems optics have become an industry standard and can provide a field-of-view in excess of 100^{o}. Currently available LCD displays do not have the resolution required to support the detailed video or graphics ultimately desired in TRIP. In response, work is progressing in-house to develop higher resolution black and white HMD's which make use of wide angle optics and flat panel CRT's. Other concepts for increased resolution and color are under consideration.

The Genie scan converter from Jovian will accept 60 Hz non-interlaced RGB signals (i.e. VGA at 640 x 480) as input and produce a 30 Hz interlaced NTSC signal as output. Monitors in most head mounted displays currently require an NTSC signal. In addition, the scan converter provides a gain adjustment and flicker filtering. Without the flicker filtering , certain horizontal lines appear to flicker and may contribute excessively to operator fatigue.

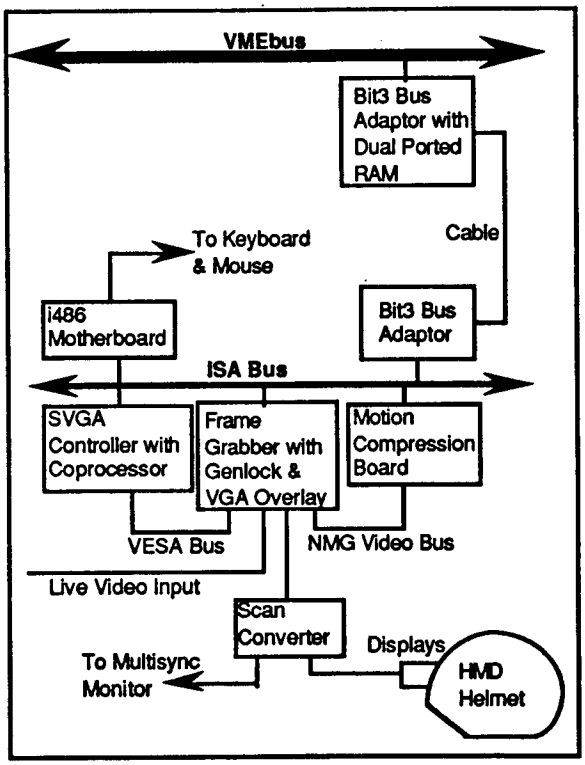


Figure 4.2.1 - Video Subsystem Block Diagram

The frame grabber and video compression boards are supplied by New Media Graphics and both include low level software drivers which are compatible with TRIP software tools. The frame grabber digitizes and scan converts live video from external cameras, allowing software manipulation of the images. In addition, this board will genlock and overlay (via graphics color keying) VGA signals using a dedicated video (VESA) bus. The video compression board enables compression and decompression of full motion (30 Hz) video using a C-Cube CL550 JPEG chip. It supports storage, playback, and network transmission of video signals using a private video bus with the frame grabber.

An Orchid Fahrenheit 1280° graphics accelerator was selected as the VGA board. It provides complete Super VGA functionality and makes use of a dedicated on-board processor to support graphics intensive applications. Low level drivers are included for the Windows operating system.

Two i486 based computers are used for subsystem control functions and as graphics engines. Each runs at a clock speed of 33 MHz, contains 8 MB of DRAM, a keyboard controller, serial ports, and an ISA bus which supports a 210 MB hard disk and subsystem video boards. They boot with a PC/AT compatible BIOS ROM and support all TRIP operating systems.

Bus adaptors are supplied by Bit3 Corporation and provide a high bandwidth link between the VMEbus and ISA bus. Boards in each bus are linked with a shielded, multiconductor cable. The VMEbus board contains 2 MB of dual ported RAM which maps into the memory space of both buses.

This subsystem accepts commands and messages from the VMEbus through the bus adaptors. Software tasks running on the i486 computers are used to control the display of live video and to generate desired graphics or text overlays. Graphics can include wireframe models driven by simulations and text may include voice menu selections or data from the remote robot. The video, graphics, and/or text are merged in the frame grabber and fed to the scan converter. This converter produces a filtered 30 Hz interlaced NTSC signal which the HMD directly accepts and displays to the operator. Future HMD's with higher resolutions may directly accept the 60 Hz non-interlaced RGB signals, allowing scan converters to be bypassed.

4.3 Audio Subsystem

Speech synthesis, environmental audio, and voice recognition are all part of the audio subsystem. Speech synthesis provides TRIP an additional path for relaying data or messages to the operator. Environmental audio can be used to supply cues or feedback on the remote environment. This can take the form of actual environmental sounds (where possible) and/or computer generated sounds which cue the operator. Voice recognition essentially replaces the keyboard as an operator input device and is required when the operator is wearing exoskeletons.

Figure 4.3.1 diagrams the audio subsystem. Components include helmet mounted headphones and microphone, a voice recognition system board, a speech synthesis board, an audio mixing system with an interface board, an i486 based ISA bus computer, and ISA to VMEbus adaptor boards.

The headphones and a microphone are part of the helmet assembly. Headphones are driven by an audio mixing system and the microphone supplies audio signals to a voice recognition system.

The voice recognition system was developed by Speech Systems Incorporated. It provides continuous speech recognition which is speaker independent and includes a large vocabulary dictionary (about 40,000 words) which can be amended by the developer. A unique combination of speech encoding, acoustic frame compression, and linguistic decoding is used to support large, variable duration segments.

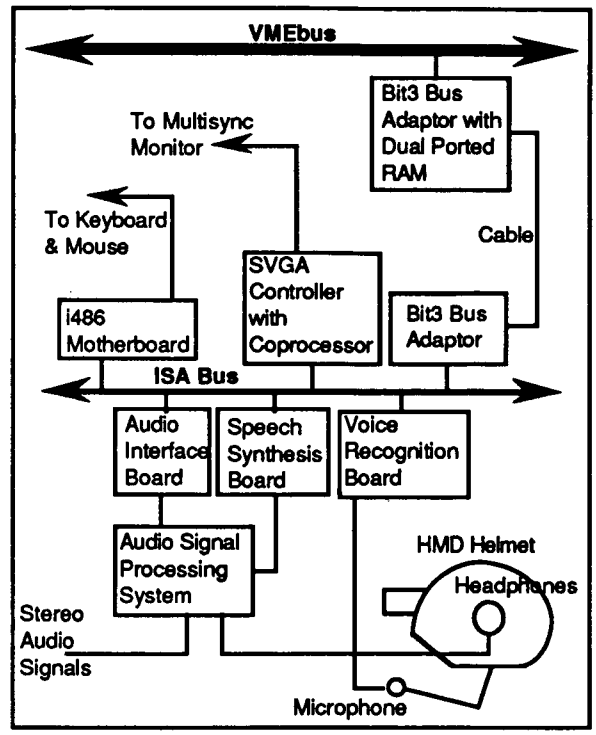


Figure 4.3.1 - Audio Subsystem Block Diagram

Speech synthesis is provided by the DoubleTalk PC board and drivers from RC Systems. The board uses its own 10 MHz, 16 bit microprocessor and supports multiple speech technologies such as text-to-speech, LPC, PCM, ADPCM, and CVSD. The analog output can directly drive headphones or be directed through a mixing system.

Audio switching, mixing, and signal processing is accommodated with a CDPC multimedia system by Media Vision. The system is based on the electronics of their Pro AudioSpectrum 16 and includes multiple audio input and output signal options. Signal processing includes digital filtering, tone control, bass enhancement, and signal equalization. The analog mixer supports volume control of each source, fade in/out, and audio panning. The system includes an ISA bus interface board and low level drivers which are compatible with TRIP software tools.

An i486 based computer is used for subsystem processing and control. It runs at a clock speed of 33 MHz, contains 8 MB of DRAM, a keyboard controller, serial ports, and an ISA bus which supports a 210 MB hard disk and subsystem audio boards. It boots with a PC/AT compatible BIOS ROM and supports all TRIP operating systems.

A Bit3 bus adaptor provides a high bandwidth link between the VMEbus and this subsystem. Boards in each bus are linked with a shielded, multiconductor cable. The VMEbus board contains 2 MB of dual ported RAM which maps into the memory space of both buses.

The audio subsystem exchanges commands and messages with the VMEbus through its bus adaptor. Software tasks running on the i486 computer handle commands for speech synthesis and environmental audio functions. Synthetic speech and environmental audio signals are processed and mixed by the CDPC system, then fed to headphones in the helmet. The operators voice is picked up by the microphone and fed to the voice recognition system for interpretation. Resulting commands are routed to the message areas of appropriate subsystems.

4.4 Communication Subsystem

The communication subsystem provides full duplex data transfers between TRIP and the remote robot using an ethernet based network. Data can represent commands, sensory information, event messages, or requests for information.

Hardware for this subsystem is shown in figure 4.4.1. It basically consists of an embedded i486 computer from Radisys and an ethernet communications board.

The i486 runs at a clock speed of 33 MHz, contains 8 MB of DRAM, a keyboard controller, serial ports, and an ISA compatible private bus which supports a 210 MB hard disk and a super VGA controller board. It is operationally similar to the i386 controller of the motion subsystem and uses the same low level drivers for VMEbus memory accesses.

The ethernet board is Western Digital (8003EB) compatible and uses commonly available packet drivers. It also supports both thin and thick ethernet cables along with TCP/IP socket libraries.

Software tasks running on the i486 computer serve three functions: (1) transfer data packets to and from the remote robot, (2) parse data packets and route information to other TRIP subsystems, and (3) handle commands from TRIP subsystems and build data packets. The data packets consists of structures which organize messages, commands, and information into a form which TRIP and the remote robot can both understand and easily parse. Future plans include incorporation of TelRIP software developed at Rice University. TelRIP (TeleRobotic Interconnection Protocol) is a layer built on top of TCP/IP with characteristics specific to teleoperation of robots. Other software tasks running on the i486 route messages to other TRIP subsystems.

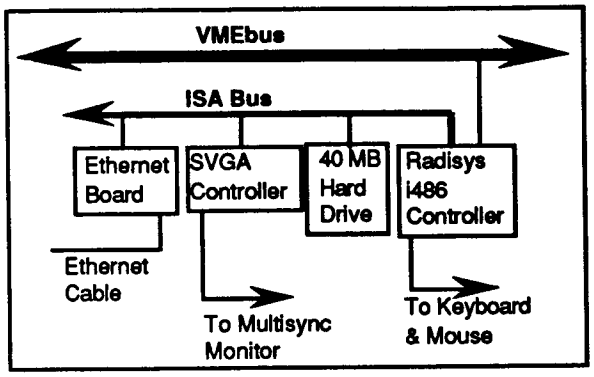


Figure 4.4.1 - Communication & Command/Control Subsystem Block Diagram

4.5 Command & Control Subsystem

The command and control subsystem coordinates interactions of all subsystems with one another. It also serves as the focal point for system configuration and subsystem calibration efforts.

This subsystem primarily consists of software, but shares the embedded i486 used by the communication subsystem shown in figure 4.4.1.

Software running on the i486 computer provides system level arbitration of subsystem task and data interactions. These interactions may be defined in terms of the active communication paths between subsystems and the messages or commands understood on those paths. In addition, each of the other subsystems may be calibrated or adjusted from here to correspond to systems on the remote robot. Examples of this include mapping of exoskeleton joints to robot joints, definition of joint limits, activation of video targeting functions, and selection of environmental audio convolution methods.

5.0 CLOSURE

The project described in this paper is primarily a system integration effort. Architectures and approaches discussed are driven by a combination of operational needs, available technologies, and flexibility to incorporate projected technologies. Design goals included ease of use, configurational flexibility, and the inclusion of growth paths. These goals were constrained by cost and performance limits.

5.1 Current Status

All the basic hardware elements of TRIP are currently being integrated. Software tasks are either in the

detailed design or implementation phase of development. The DART anthropomorphic target robot is currently in the implementation phase and will be interfaced to TRIP by the end of this year. The AERCAM free-flying target robot is in the detailed design phase of development.

5.2 Future Work

Future work will address the implementation and testing of newly developed subsystem and programming technologies. Examples in the video area include higher resolution black & white monitors, direct VGA interfaces with wide angle optics, and computational graphics models as in reference 14. Motion control examples include the addition of force/torque reflection, haptic feedback, and teleprogramming concepts as described in reference 25. TRIP will also make use of 3-D acoustics research and signal processing techniques, such as those of reference 19. Robot communications will be enhanced through updated versions of TelRIP[20,21] software. Finally, an icon or block diagram based visual environment will be used for system configuration and subsystem calibrations.

6.0 REFERENCES

TRIP System Level

[1] P.G. Backes, "Ground Remote Control for Space Station Telerobotics with Time Delay", 15th Annual AAS Guidance and Control Conference, Keystone, CO, February 1992.

[2] P.S. Schenker, A. K. Bejczy, W. S. Kim, S. Lee, "Advanced Man-Machine Interfaces and Control Architecture for Dexterous Teleoperations", IEEE Oceans '91, Underwater Robotics Session, Honolulu, HI, Oct. 1-3.

[3] T. B. Sheridan, "Musings on Telepresence and Virtual Presence", Presence - Teleoperators and Virtual Environments, Volume 1, Number 1, pp: 120-125, MIT Press, Cambridge, MA, 1992.

[4] L. C. Li, H. Nguyen, E. Sauer, "A Integrated Dexterous Robotic Testbed For Space Applications", Fifth Annual Workshop on Space Operations Applications Research, Volume 1, Telerobotics and Mechanisms Session, pp 238-245, Houston, TX, July 1991.

[5] K. J. Korane, "Sending a Robot to do a Man's Job", reprint from Machine Design, Vol. 63, Num. 22, November 7, 1991.

[6] Anon., video entitled "TOPS System Demonstration", Naval Ocean Systems Center, San Diego, CA, June 1991.

[7] Anon., video entitled "Introduction to Technology and Application", Kraft Telerobotics, Inc., Overland Park, KS, Dec. 1991.

Motion Subsystem

[8] B. A. Marcus, B. Eberman, "Exos Research on Master Controllers for Robotic Devices", Fifth Annual Workshop on Space Operations Applications Research, Volume 1, Sensing and Display Session, pp 238-245, Houston, TX, July 1991.

[9] Anon, "2D/6D Mouse Technical Reference Manual", Logitech, Inc., Fremont, CA, 1991.

Video Subsystem

[10] P. Anderson, "Virtual Reality: Is it for Real? And What's in it for Imaging?", reprint from Advanced Imaging, June 1991.

[11] E. Howlett, "Wide Angle Orthostereo", LEEP Systems/Pop-Optix Labs, Waltham, MA, Feb. 1990.

[12] E. Howlett, "Wide Angle Color Photography Method and System", U. S. Patent 4406532, Sept. 27, 1983.

[13] J. T. Carollo, "Helmet Mounted Displays", Proceedings of the International Society for Optical Engineering, SPIE Publication No. 1116, Bellingham, WA, March 1989.

[14] W. Robinett, J. P. Rolland, "A computational Model for the Stereoscopic Optics of a Head Mounted Display", Presence - Teleoperators and Virtual Environments, Volume 1, Number 1, pp: 45-62, MIT Press, Cambridge, MA, 1992.

[15] Anon, "Flight Helmet Operating Instructions and Owners Manual", Virtual Research, Sunnyvale, CA, 1992.

[16] Anon, "Super Video Windows Programmers Guide", New Media Graphics Corp., Billerica, MA, 1991.

Audio Subsystem

[17] W. S. Meisel, M. T. Anikst, S. S. Pirzadeh, J. E. Schumacher, M. C. Soares, D. J. Trawick, "The SSI Large-Vocabulary Speaker-Independent Continuous Speech Recognition System", IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 1, pp: 337-340, Toronto, Ontario, Canada, May 1991.

[18] W. S. Meisel, M. T. Anikst, "Efficient Representation of Speech for Recognition", reprint from Speech Technology, Feb./March 1991.

[19] E. M. Wenzel, "Localization in Virtual Acoustic Displays", Presence - Teleoperators and Virtual Environments, Volume 1, Number 1, pp: 80-103, MIT Press, Cambridge, MA, 1992.

Communication Subsystem

[20] J. D. Wise, L. Ciscon, "TeleRobotics Interconnection Protocol, Version 1.5", Universities Space Automation / Robotics Consortium, Technical Report #9103, Dept. of Electrical and Computer Engineering, Rice University, Houston, May 1991.

[21] J. D. Wise, L. Ciscon, presentation notes from the "First Annual TelRIP Workshop", Johnson Space Center, Houston, TX, April 1992.

[22] S. G. Kochcan, P. H. Wood, "Unix Networking", Hayden Books, Howard Sams & Co., First Edition, Carmel, IN, 1990.

[23] Anon, "Distinct TCP/IP for Microsoft Windows - Software Development Kit", Distinct Corp., Saratoga, CA, 1992.

Command & Control Subsystem

[24] C. H. Small, "Real-Time Operating Systems", EDN, Jan. 7 issue, pp: 115-138, Denver, CO, 1988.

[25] J. Funda, T.S. Lindsay, R. P. Paul, "Teleprogramming: Toward Delay Invariant Remote Manipulation", Presence - Teleoperators and Virtual Environments, Volume 1, Number 1, pp: 29-44, MIT Press, Cambridge, MA, 1992.

[26] C. Petzold, "The Visual Development Environment - More Than Just a Pretty Face?", PC Magazine, June 16, Vol. II, Num 11, pp:195-237, Boulder, CO, 1992.

[27] Anon, "Microsoft Windows NT Operating System", Data Sheet, Microsoft Corp., Redmond, WA, 1992.

[28] Anon, "Microsoft Windows Operating System Version 3.1 - An Overview of New Features", Corporate Backgrounder, Microsoft Corp., Redmond, WA, 1992.

[29] C. H. Small, "Windows and Engineering Software", EDN, April 9 issue, pp:123-132, Denver, CO, 1992.

[30] W. D. Peterson, "The VMEbus Handbook", 2nd Edition, VFEA International Trade Association, Scottsdale, AZ, 1991.

[31] Anon, "EPConnect Software Manuals", Radisys Corp., Beaverton, OR, 1990.

[32] Anon, "Real Time Operating System Runs Windows", reprint from Microcomputer Solutions, May/June 1991.

[33] C. Carvallio, "Adding DOS Software to Real-Time Systems: New Strategies for Real World Control", Intel Corp., Industrial Computer Div., Hillsboro, OR, 1991.

N93-32108

INTERACTIVE AND COOPERATIVE SENSING AND CONROL FOR ADVANCED TELEOPERATION

Sukhan Lee

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91011

ABSTRACT

This paper presents the paradigm of interactive and cooperative sensing and control as a fundamental mechanism of integrating and fusing the strengths of man and machine for advanced teleoperation. The interactive and cooperative sensing and control is considered as an extended and generalized form of traded and shared control. The emphasis of interactive and cooperative sensing and control is given to the distribution of mutually non-exclusive subtasks to man and machine, the interactive invocation of subtasks under the man/machine symbiotic relationship, and the fusion of information and decision-making between man and machine according to their confidence measures. The proposed interactive and cooperative sensing and control system is composed of such major functional blocks as the logical sensor system, the sensor-based local autonomy, the virtual environment formation, and the cooperative decision-making between man and machine. The Sensing-Knowledge-Command (SKC) fusion network is proposed as a fundamental architecture for implementing cooperative and interactive sensing and control. Simulation results are shown.

INTRODUCTION

Early attempts on teleoperation were based on tight coupling between the manipulator and the operator through mechanical linkages or steel tapes, as is the case of the AEC Argonne Laboratory series [1], or electrical or hydraulic connections, as is the case of the GE telemanipulators built by Mosher[2].

The telemanipulation based on the direct coupling between man and machine severely limits its performance : it neither accommodates the desirable mechanical dexterity due to the difficulty of manually coordinating multiple joints, nor allows high task complexity due to the difficulty of achieving required compliance. It gives an excessive burden on the operator, which may cause long task completion time with a high failure rate.

The need to improve mechanical dexterity in teleoperation and achieve desirable compliance during teleoperation, so as to deal with more complex tasks under a partially constraint environment but with the comfort of human operator, has prompted the development of the following teleoperation paradigms :

1) The generalized bilateral telemanipulation [3, 4, 5, 6] in which the tight and one-directional coupling between the master and the slave is replaced by loose and two-directional coupling characterized by computer-based bilateral information transformation and exchange. This allows that the slave arm may not need to be the exact kinematic replica of the master arm, and that the operator can feel the contact force felt by the slave arm through force feedbacks, which allows human to execute compliance control.

2) The supervisory control with shared and traded control [7, 8], in which a task is decomposed into temporarily (traded control) or spatially (shared control) disjoint subtasks that are to be distributed to man and machine. For instance, the operator can be supported by software jigs or spatial support means[9, 10] which take advantages of spatial constraints in the task to allow the slave manipulator to control those degrees of freedom specified by the motion constraint, while the operator controls the rest of degrees of freedom. Or, the slave arm with force/torque sensors is responsible for automatic compliance control, while the operator is responsible for the motion control. The supervision of telemanipulation[11] is done by the supervisory loop closed through the human operator, for which visual and graphic displays and force reflections from the remote site play an important role.

The recent advancement in the theory and practice of robotics and intelligent systems makes it necessary to exploit new generation of teleoperation which fully utilizes the high degree of mechanical dexterity provided by redundant and multiple arms and the capability of a robot performing sensor-based local autonomy. Especially, the role of man and machine should be redefined for advanced teleoperation in such a way that the slave arm becomes an active partner of the human operator, supporting perception, decision-making, and cooperative task execution. To achieve this requires to explore a fundamental mechanism of integrating and fusing the strengths of man and machine for advanced teleoperation.

This paper presents a paradigm of interactive and cooperative sensing and control as the fundamental mechanism of integrating and fusing the strengths of man and machine for advanced teleoperation. The interactive and cooperative sensing and control is considered as an extended and generalized form of traded and shred control. The emphasis of interactive and cooperative sensing and control is given to the distribution of mutually nonexclusive subtasks to man and machine, the interactive invocation of subtasks to achieve the man/machine symbiotic relationship, and the fusion of information and decision-making between man and machine according to their confidence measures.

THEORY OF INTERACTIVE AND COOPERATIVE SENSING AND CONTROL

The quality of teleoperation depends on the performance of the operator in perceiving and understanding task mechanisms correctly and in generating control commands precisely in consistency with his/her perception and intention. The quality of teleoperation also depends on the performance of the machine (as a master-slave system) in providing accurate and sensitive control which is stable and robust under disturbances, system nonlinearities, and time delays.

As a means of enhancing the performance of the operator, there have been developed methods for accomplishing powerful telepresence based on sensory feedbacks using visual displays and force reflections, as well as methods for effectively training the operator to achieve the high level of expertise. On the other hand, the development of advanced teleoperator controllers based on the concept of impedance, passivity, dynamic coordination, and predictive modeling has been pursued as a means of improving the performance of the machine.

However, there exist fundamental limitations for the operator to achieve accurate perception of task geometries and control behaviors and, even more so, to accomplish precise coordination between perception and action. This is mainly due to the impreciseness and low bandwidth in human sensory-motor coordination: human depends heavily on sensor-based adaptive motion corrections to compensate for imprecise positioning and is unable to respond to high bandwidth tasks. And, partly due to the difficulty of implementing powerful telepresence as well as high performance of control.

The best way of relaxing the above limitations is to fully utilize the strengths of man and machine in such a way as to achieve the mutual compensation of individual weaknesses. The strength of human lies in understanding task mechanisms, recognizing objects, generating task and motion plans under global constraints, whereas the strength of machine lies in precision positioning, quantization of primitive features, repetition of memorized tasks, and sensor-based local reflex. Attempts have been made to incorporate the strengths of man and machine in teleoperation: traded control temporally decompose a task and assigned to human and machine according to whether human or machine fits for a give subtask, while shared control spatially decompose a task into subtasks to be carried out by man and machine simultaneously. An instance of shared control is that compliance control is automatically accomplished by machine based on sensed forces, while position control is done through operator's manual control.

Although traded control and shared control provide a means of combining the strengths of man and machine, they do not present a general and powerful methodology of integrating man and machine. This is because traded and shared control is based on clear-cut decomposition of tasks into subtasks to be distributed individually to man and machine, where such decomposition is often difficult to achieve, resulting in overly simplified distribution of a task. More importantly, such a clear-cut decomposition eliminates the possibility of fusing multiple sources of information and decision-making from man and machine.

We propose interactive and cooperative sensing and control as a fundamental paradigm of integrating and fusing the strengths of man and machine for teleoperation. The interactive and cooperative sensing and control is an extended and generalized form of traded and shared control. The emphasis of interactive and cooperative sensing and control is given to the distribution of mutually nonexclusive subtasks to man and machine, the interactive invocation of subtasks with symbiotic relationship, and the fusion of information and decision-making from man

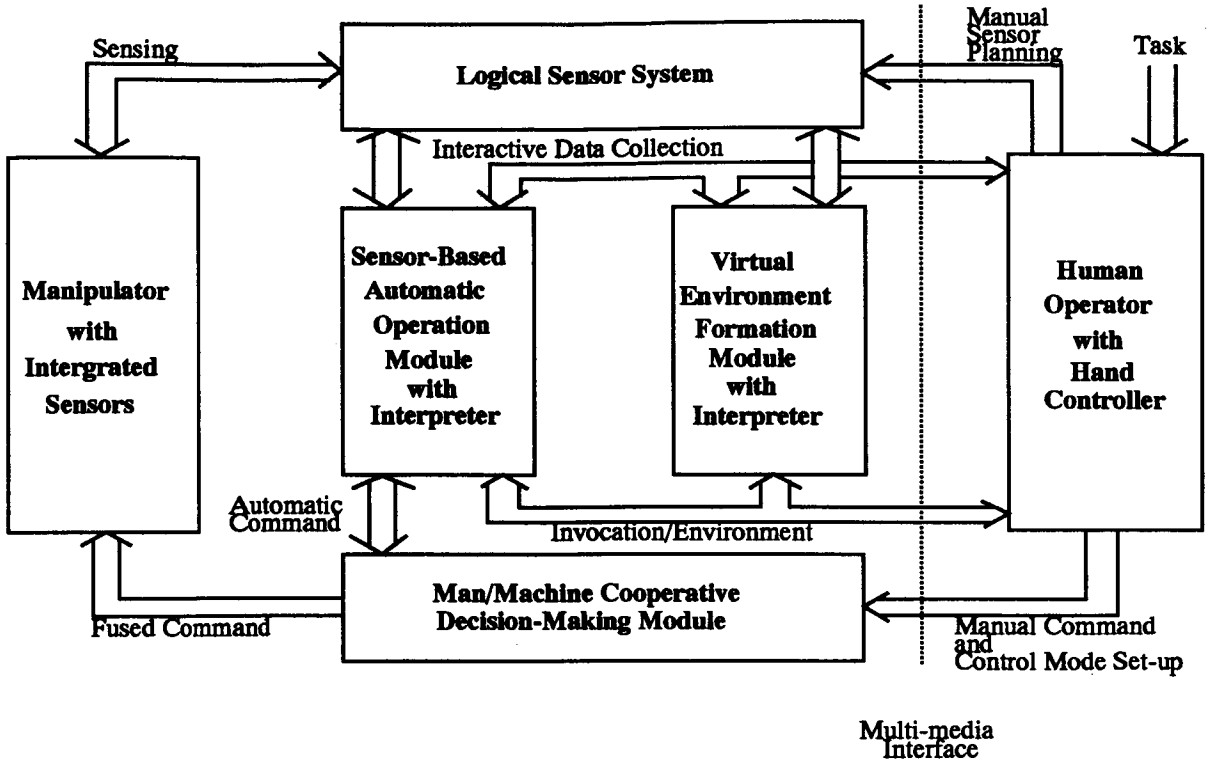


Fig.1 Interactive and Cooperative Sensing and Control for Advanced Teleoperation.

and machine according to the degree of their confidence.

The interactive and cooperative sensing and control consists of the following major functional blocks : 1) logical sensor system. 2) sensor-based local autonomy. 3) virtual environment formation. 4) cooperative decision-making between man and machine.

Logical Sensor System

A logical sensor represents, in an abstract form, one of the many functional capabilities that the integrated sensor system can provide. The distance sensor, the surface orientation sensor, the force/torque sensor, the feature finding sensor, etc. are a few examples of logical sensors. There may or may not exist a direct association between a logical sensor and a physical sensor, such that a logical sensor can achieve its goal (to generate its output) based on the outputs of other logical sensors and/or physical sensors. Logical sensors can be hierarchically organized into a logical sensor system based on their functional interdependency. A logical sensor system not only provides a symbolic list of the various perceptual capabilities of a robot, but also represents a number of different ways of accomplishing the goal of a logical sensor. The latter is especially useful for sensor fusion. The symbolic representation of a logical sensor system provides an effective tool for the intelligent interface with the operator performing interactive and cooperative sensing. For instance, a logical sensor can be invoked by the operator in response to the system's request for providing sufficient information for a sensor-based automatic operation or a virtual environment formation initiated by the operator.

Sensor-Based Automatic Operations

Sensor-based automatic operations are for providing the manipulator with the capability of local autonomy, such that man/machine cooperative control can be accomplished. A list of sensor-based automatic operations are predefined, out of which the operator can select and invoke a desired sensor-based automatic operation. Examples of sensor-based automatic operations include automatic tracking, automatic centering, automatic aligning, automatic compliance, etc. Once invoked, it is sent to the interpreter to transform it into a sequence of actions executable by the manipulator; during the process of interpretation, the interpreter automatically inquires the logical sensor system and/or the operator for the information necessary for the complete specification of the corresponding sensor-based operation. The operator performs, if necessary, sensor planning and interactive sensing, and invokes logical sensors.

Virtual Environment Formation

Virtual environment formations are for providing the operator with an artificially created environment (called virtual environment) which enhance the operator's understanding of control environment and task mechanism, and, consequently, improve the fidelity of operator's manual control. The generated virtual environment provides a guidance and assistance for operator's manual control. A list of virtual environment operations are predefined, out of which the operator can select and invoke a desired virtual environment operation. Virtual environment operations generates displays or reflects forces which partially or fully inform the operator of the task specifications obtained by logical sensors for sensor-based automatic operations, or provide sensory feedback indicating the discrepancy from the sensor-based automatic operation. Examples include the surface normal display, the virtual force field in free space, the display of desired end effector orientations, etc. As is done for sensor-based operations, once invoked, it is sent to the interpreter to transform it into a detailed sequence of operations with interactive information collection. A virtual environment formation may or may not accompany with the corresponding sensor-based automatic operation.

Man/Machine Cooperative Decision-making

Since it is allowed that sensor-based automatic operations and operator's manual operations carry out mutually non-exclusive tasks, we need to provide a mechanism for fusing two different source of decisions, or, simply decision fusion. The degree that individual decisions contribute to the final (optimal) decision should depend on their credibility. The credibility of decision by the machine can be estimated in terms of the certainties involved in the sensor measurements, the decision-making rules, and the constraints used in the decision making. Whereas, the credibility of decision by the operator depends on the level of expertise obtained by the experience. However, it should be noted that such credibilities are subject to variation not only with respect to time but also with respect to control situations: For instance, in case a jamming situation occurred in the peg-hole insertion process, the operator's capability of making an error correction operation based on a global planning may be more dependable than the solution based on the sensor-based automatic insertion process. To handle this variations, the operator is allowed to set the degree of contribution of individual decisions heuristically.

Information Flow

Fig1. illustrates the information flow between the major functional blocks of the interactive and cooperative sensing and control system. The information flow can be summarized as follows:

1) Given a task, the operator may invoke the sensor-based automatic operation and/or virtual environment formation, by selecting a menu from the prespecified lists.

2) The operator can also select the system control mode as manual control, shared control, cooperative control, or automatic control, by simply adjusting the relative weight between the sensor-based automatic operation and the manual operation in cooperative decision-making. It should be noted that the sensor-based automatic operations can be used solely for the purpose of virtual environment formation, without participating in cooperative control, in case the operator invokes both the sensor-based automatic operation and the virtual environment formation, but assigns zero weight to the sensor-based automatic operation in cooperative decision-making.

3) Prior to the invocation of the sensor-based automatic operation module or the virtual environment formation module, the operator may need to perform sensor planning to ensure that the invoked operation can retrieve correct information from the logical sensor system. The interpreter of the sensor-based automatic operation or the virtual environment formation generates executable commands by filling out the existing templates through the interaction with the logical sensor system and/or the operator.

4) The virtual environment formation module provides the operator with the information representing the current control situation, especially in terms of the deviation of manual control from the sensor-based automatic operation, based on the multi-media interface using graphic displays, Cartesian space force fields at the operator's hand, and sound. The virtual environment formation offers, among other things, the visual servoing guidance and the virtual compliance which keep the manipulator from moving away from the desirable pose.

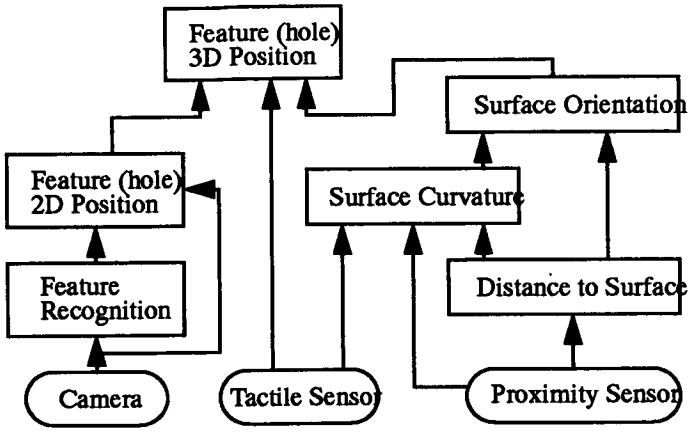


Fig.2 A part of the logical sensor system chosen for illustration

Scenario

To explain the above concept in more detail, a typical scenario of interactive and cooperative sensing and control for advanced teleoperation is described in the following based on the peg-hole insertion task:

- Let us assume that the manipulator in a remote site has various sensors such as proximity sensors, force/torque sensors, tactile sensors, and a mini-camera mounted on the end effector, as well as stereo cameras fixed in space for the purpose of globally monitoring the task space. The capabilities of the above sensors can be summarized and organized in a logical sensor system, e.g., as shown in Fig2. Each logical sensor has its own sensing goal to be achieved through the logical sensor hierarchy. The data that a logical sensor represents is associated with a confidence measure to be used in sensor fusion, which may occur when multiple paths of achieving the sensing goal exists in the logical sensor hierarchy, and in cooperative decision-making.

- With the aid of the various sensors mounted on the end effector, the manipulator is able to perform various simple sensor-based automatic operations: maintaining orientations, tracking predefined features, reaching identified positions, reacting to contact forces for compliance, centering on a geometric feature, aligning to a surface normal, etc. These sensor-based automatic operation primitives require a minimal operator intervention for interpretation. For instance, the "Align Surface Normal" primitive requires the operator to position the end-effector near the corresponding surface prior to the invocation of the primitive. The executable command will then be automatically generated by the interpreter filling out the corresponding template through the interaction with the logical sensor system, and/or the human operator.

- Let us also assume that the system is capable of providing the operator with virtual environments based on visual displays using video images and graphics, 3D force field at the operator's hand, and sound. The virtual environment can be formed by representing the discrepancy between the sensor-based automatic operation and the operator's manual operation. In fact, a sensor-based automatic operation can be invoked solely for the purpose of virtual environment formation, should the operator desired to do so. Other list of virtual environment include a force field about surface normals, a graphic overlay of commanded manipulator configuration on the video image, a graphic display of contact force and moment, etc.

- Now, let us consider that the operator is given a peg and hole insertion task, where the hole is assumed to have very small tolerance. The major difficulty of the above peg-hole teleoperation lies in the operator's generation of accurate peg motion with correct peg orientation and position. Especially, maintaining correct peg orientations throughout the insertion process is considered vital for avoiding jamming, but often not so easy to be achieved by the human operator.

- Thus, the operator can invoke "Align Surface Normal" for a sensor-based automatic operation as well as a virtual environment formation, so that not only the force field about the surface nor-

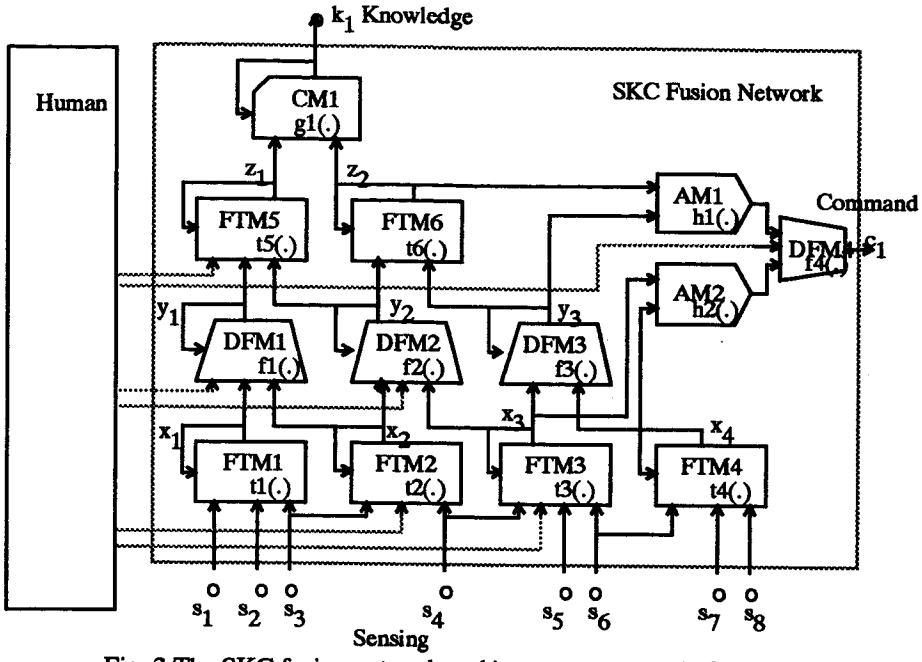


Fig. 3 The SKC fusion network architecture composed of 4 different modules: FTM, DFM, CM and AM

mal, generated by the virtual environment formation module, guides the orientation of the operator's motion, but also control decision is shared by the operator and the sensor-based automatic operation module according to their strengths.

SENSING-KNOWLEDGE-COMMAND FUSION

The mechanism of sensor data fusion[12,13,14,15,16,17] can provide a fundamental means for achieving system integration since it combines multiple uncertain sensor data into more accurate and reliable estimates, identifies faulty sensors through consensus verification, and maintains consistency with existing constraints. We extend the notion of "sensor data fusion" toward a more general concept of "Sensing-Knowledge-Command(SKC) fusion" to include the integration of feature transformation and abstraction, data and concept fusion, knowledge propagation for consistency satisfaction and cooperative planning and decision-making.

The "SKC fusion network" provides a fundamental architecture for implementing cooperative and interactive sensing and control for advanced teleoperation system[18]. The SKC fusion network establishes the mechanism of achieving network consistency in real-time through dynamic evolution of network states: once invoked by inputs or stimuli, the SKC fusion process enforces the network to converge to new equilibrium states through the network dynamics of data fusion, feature transformation, and constraint propagation. The cooperative control of man/machine systems is then accomplished through the SKC fusion process invoked by stimuli from both human and machine, where sensing, knowledge, and command of a human and a machine are tapped into the network to provide inputs or stimuli for the network.

SKC Fusion Network

"SKC fusion network" represents a fundamental robotic architecture based on which the real-time connection between perception and action is accomplished. The SKC fusion network is formed by the interconnection of four basic modules: the data fusion module, the feature transformation module, the constraint module, and the action module, as shown in Fig. 3. A data fusion module(DFM) takes one or more data representing an object feature and produces the optimal estimate for the feature in cooperation with the initial state of the module. A feature transformation module(FTM) extracts a primitive features from the raw sensory data or transforms a set of primitive features into the more abstract, higher level features. An action module(AM), as a special case of a feature transformation module, issues the command to the environment based on the predefined laws triggered by a set of features. A constraint module(CM) represents system knowledge which put a constraint upon a set of feature values associated with the knowledge: the feature values

should be adjusted in such a way as to achieve a maximum consistency with the associated knowledge. The output of each module indicates the current estimates of the corresponding feature or knowledge, and is kept as the current state of the module. The state transition of a module propagates in both directions (forward and backward), and invokes the state transition of other modules having functional relationship with it. In this sense, the interconnection among modules is considered bidirectional, as represented in Fig. 3 by a feedback loop associated with each module. The domain knowledge is embedded in the network in two ways: explicitly by the constraint module, and implicitly by the functions of feature transformation modules as well as the network structure.

Network Dynamics

The mechanism of SKC fusion network can be interpreted in terms of two operational modes: the forward mode and the backward mode. The forward mode first extracts primitive features from sensor data through low level feature transformation modules, and subsequently produces more abstract form of features through higher-level feature transformation modules. The forward mode also allows the data and concept fusion to occur through data fusion modules, whenever multiple and redundant data are available for a single feature or concept. The backward mode starts to operate upon the activation of a constraint module: based on the error detected at the constraint module, all the feature values connected to that constraint module are adjusted to satisfy consistency. The new updated feature values (as the output of feature transformation modules) in turn invokes the adjustment of lower level features connected to the module. Through a cycle of forward and backward information propagations, the network reaches an equilibrium state, i.e., all the features and concepts have consistent estimates which are optimal in the sense that redundant sources of information are fused under the constraints provided by system knowledge.

The entities of the SKC fusion network, such as data, features, concepts, and knowledge, are represented by their nominal values or equations and the degree of uncertainties associated with the normal values or equations. Thus, during a cycle of forward and backward information propagations, not only the nominal values or equations but also the degree of their uncertainties need to be adjusted. Probabilistic modeling and inference can provide a means of achieving the adjustment of the nominal values or equations, and the degree of their uncertainties. For instance, in the forward process, the output of a FTM can be characterized by a random variable, x,

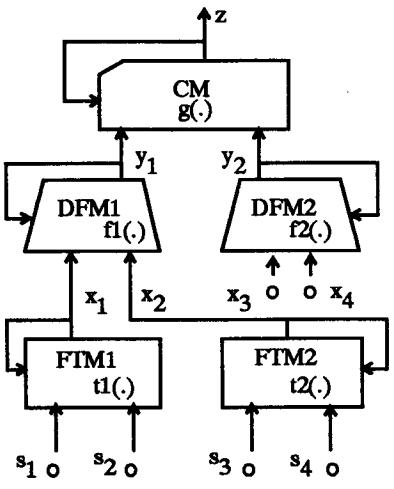


Fig. 4 A simple SKC Fusion Network used for the Description of Network Dynamics

where the probability density function, p(x), of x is determined based on the input random variable, s, of a known probability density function and the corresponding feature transformation function, x = t(s). The output, y, of a DFM can be determined based on the maximum likelihood estimate, the successive Bayes estimate, and the minimum variance estimate.The backward process for a CM or a FTM can be accomplished by the nonlinear optimization or the inverse mapping paradigm based on input update rule. The backward process for a DFM can be accomplished simply by the direct propagation of the output to individual inputs. The problem associated with the above approach based on successive computation of forward and backward propagation is that it is not suitable for real-time implementation due to the computational complexity involved in the processes, as well as the difficulty of processing non-Gaussian signals generated by non-linear transformations. Therefore, in this paper, we present a new approach for accomplishing forward and backward processes of individual modules simultaneously and concurrently, based on the dynamic evolution of the states of modules. The approach is based on representing the SKC network as a dynamic system in which the network dynamic state is evolving toward the equilibrium state, once invoked by input stimuli, as described in more detail in the following.

Dynamics of SKC Fusion Network

For the clear description of the concept of dynamic evolution of the SKC fusion network, let us consider a simple SKC fusion network illustrated in Fig. 4. Let us assume that, upon the stimuli given to the network, FTM1 and FTM2 have new inputs s_1 and s_2, and s_3 and s_4, respectively. This involves the states of individual modules simultaneously evolve toward the new equilib-

rium states that maintains network consistency. We propose that the evolution of system states is governed by the following dynamics:

1) The DFM-CM dynamics:

$$y_1 = -\lambda_{y1}\{y_1 - f1(x_1^t, x_2^t)\} - \gamma_{y1}\nabla\phi\{z^0 - g(y_1, y_2)\} \quad (1)$$

2)the FTM-DFM dynamics:

$$x_1 = -\lambda_{x1}\{x_1 - t1(s_1, s_2)\} - \gamma_{x1}\nabla\phi\{y_1 - f1(x_1, x_2)\} \quad (2)$$

$$x_2 = -\lambda_{x2}\{x_2 - t2(s_3, s_4)\} - \gamma_{x2}\nabla\phi\{y_1 - f1(x_1, x_2)\}$$

where the initial conditions are given by the equilibrium states.

(1) and (2) represent a set of fundamental dynamic equations governing the behavior of the SKC fusion network in reaching a new network equilibrium state. The first term of a dynamic equation represents the forward process, whereas the second term represents the backward process. To deal with more complex networks, we need to simply repeat the same form of dynamic equation used for (1) and (2) for individual module of network with the proper assignment of module functions and coefficients.

The variation of y_1 due to the forward process and the variation of y_1 due to the backward process should be determined in terms of the uncertainty associated with the forward process and the backward process. These variations can be controlled by the ratio between the coefficients, λ_{y1}, γ_{y1}, γ_{x1} and γ_{x2}. It is possible that the above dynamic coefficients can be assigned in such a way that the result of dynamic evaluation approximately matches the result from a probabilistic model. In fact, the above dynamic equations can be considered as a general form of the minimum variance estimate described previously. This observation allows not only to obtain the optimal dynamic coefficients but also to update the uncertainties (represented by covariance matrices) involved in individual states.

SIMULATION

To demonstrate the operation of the SKC fusion network based on network dynamics given by (1) and (2), we chose the following simple example. A robot is given a task to pick up a right triangle among many different shapes of triangles. The robot is assumed to have two logical sensors:[4] one for the measurement of edge normals (called the "edge-normal sensor") and the other for the measurement of internal angles (called the "angle sensor"). The angle sensor is easier to handle, but has more uncertainty than the edge-normal sensor. For a given triangle, the robot measures each angle twice with the angle sensor and takes the average of two measurements. On the other hand, the robot measures the edge normals with the edge-normal sensor, and computes the internal angles from the measured edge normals. Then, a decision

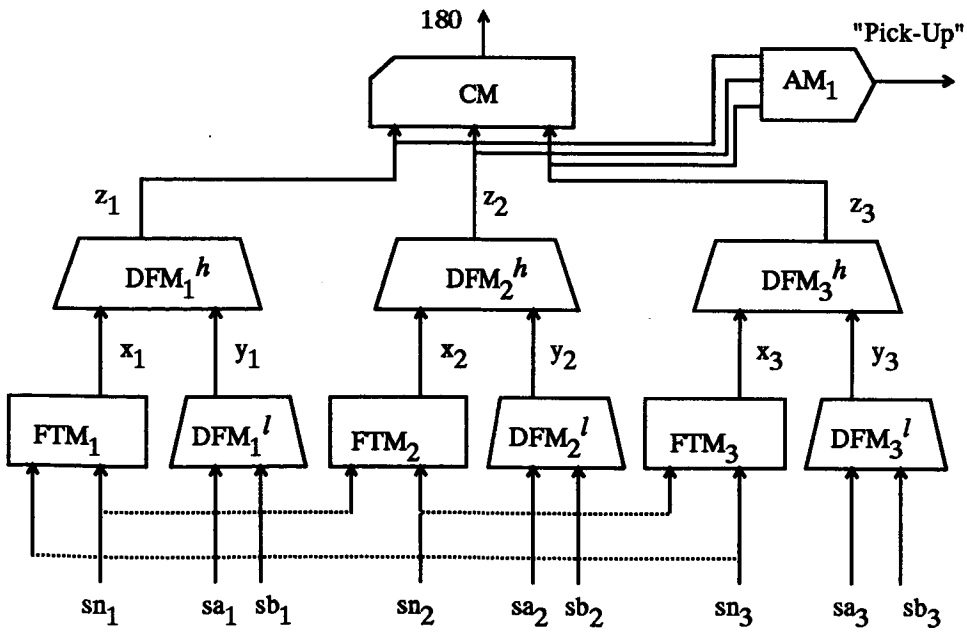


Fig.4 The SKC Fusion Network for the Task to Pick-Up the Right Triangle

is made on the size of each angle based on data from both sensors. If any of three angles is a right angle, the "pick-up" command is issued. The sum of three internal angles is subject to be 180 degrees. It is assumed that the sensor data have the known, independent Gaussian distributions.

For the task described above, we can organize the SKC fusion network as shown in Fig. 4. The two sensed data from the angle sensor, sa_1 and sb_1, are fused into y_i, through DFM_i^l. FTM_i takes the edge normals sn_i as its input, and computes the internal angle, x_i, between those two edges. Through the feature transformation equation, these x_i and y_i are in turn fused into the angle estimates, z_i, through the higher level DFM_i^h. CM checks if z_i satisfy the constraint, i.e., the sum of internal angles is 180 degrees. As mentioned in section 3, if an error is detected in CM, it is used to adjust z_i, and propagated backward to adjust x_i and y_i. The variances of x_i, y_i, and z_i can be computed from those of sensor data, based on the assumption of the independent, Gaussian distribution. The feature transformation function in FTM_i, $t_i(.)$ is defined as:

$$t_i(sn) = (180 - (sn_i - sn_j)) \text{ MOD } 360)$$

where sn_i and sn_j are two edge normals, and we simply use the averaging function as the data fusion function, $f_i(.)$, i.e.,

$$f_i(\mathbf{s}) = (\Sigma_{j=1,M}\ s_j)/M$$

where **s** is the input vector with dimension M.

Based on (1) and (2), the dynamic equation for z_i is formulated as:

$$z_i = -\tau_{zi}\, z_i + (x_i+y_i)/2 + \sigma_{zi} * (180 - \Sigma_{j=1,3}\, z_j), \quad \text{for } i=1,2,3.$$

The second term of the right hand side comes from the forward process through DFM_i, whereas the third term from the error in CM. Since the standard deviation σ_{zi} is used for the coefficient of the CM output error, with a smaller variance of z_i is less affected by the output error of CM. Similarly, the dynamic equations for x_i and y_i are formulated as:

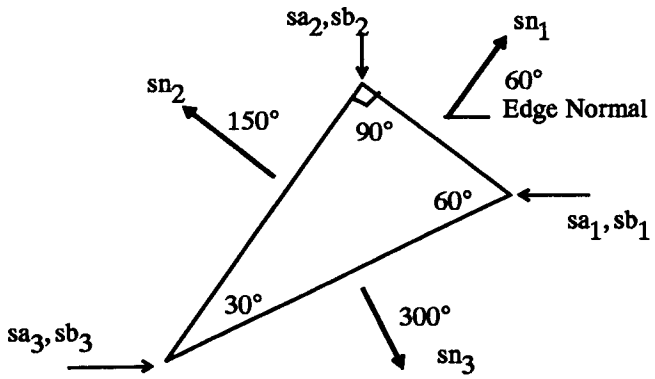


Fig. 5 The Sample Triangle and the sensing location

$$x_i = -\tau_{xi}\, x_i + ((180 - (sn_i - sn_j)) \text{ MOD } 360) + \sigma_{xi} * (z_i - x_i), \quad \text{for } i=1,2,3$$

$$y_i = -\tau_{yi}\, y_i + (sa_i + sb_j)/2 + \sigma_{yi} * (z_i - y_i), \quad \text{for } i=1,2,3$$

Note that the error is defined here as the difference of z_i from y_i and x_i, since the input and output of DFM should be equal when an equilibrium state is reached.

Although the SKC fusion network shows the different result according to the input, a typical result for a triangle in Fig. 5 is shown in Fig. 6 with the sensor statistics and data in Tab. 1 The initial equilibrium state is chosen as the ideal data for the equilateral triangle in which all the **x**, **y**, and **z** are [60.0 60.0 60.0] without any errors in CM and DFMs1. The sensed edge-normal data vector, **sn**, is quite accurate due to its small variance while the sensed angle data vector deviates a lot from the actual data. Starting from the initial equilibrium state, the top level estimate vector of angles, **x**, converges to the equilibrium states, $\mathbf{x}_e$ = [59.6 89.4 31.5], which is very close to the real accurate value, [60.0 90.0 30.0] as shown in a). The error between **x** and the real value decreases gradually and finally converges to a small value as shown in b)

The error in CM remains as almost zero all the time as shown in c), and the errors in DFMs grow during the transient state, and then converge to near zero as shown in c) and d). Note that **x** and **z** are almost same in the new equilibrium state while the deviation of **y** from **z** is big. This is because there exist considerable errors in the data from the angle sensor, but y has been adjusted toward the real value in the new equilibrium state.

To explore the effect of the backward process, the above simulation is repeated with change that the backward process is invoked at time t=2, and its results are shown

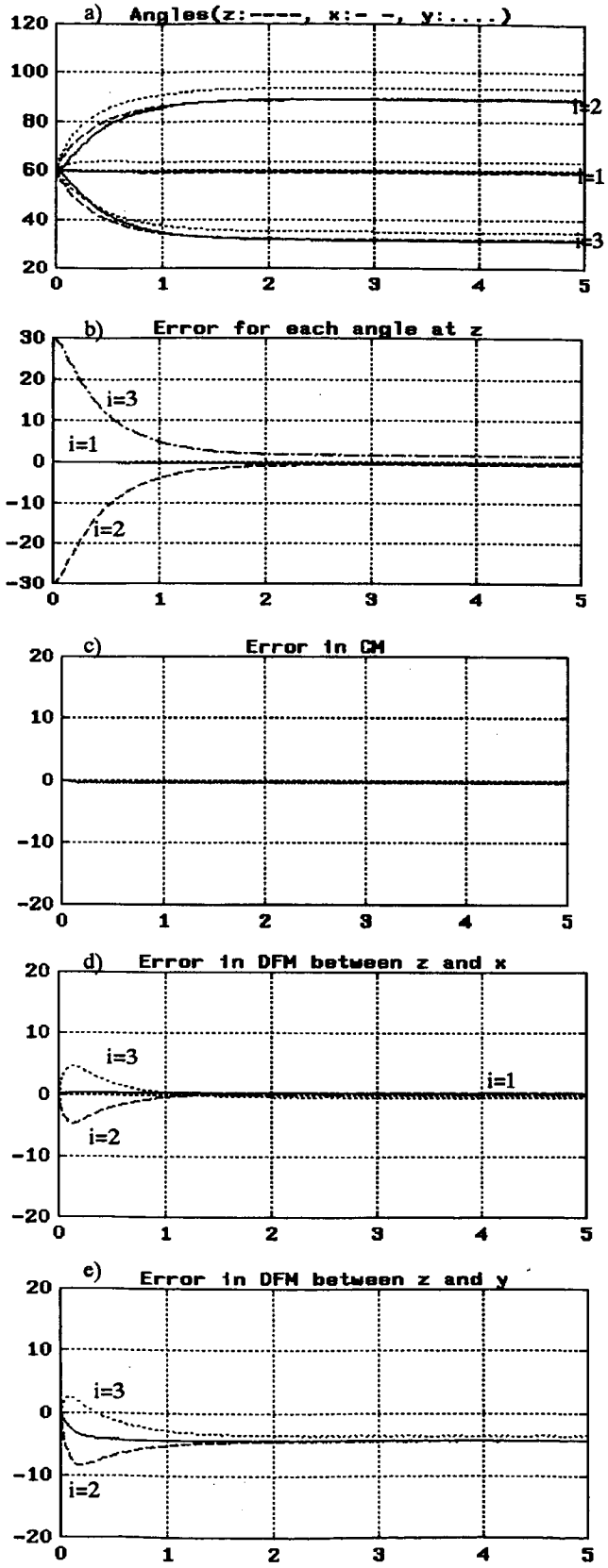


Fig. 6 Simulation results for the triangle in Fig. 5 with the parameter in Tab.1.

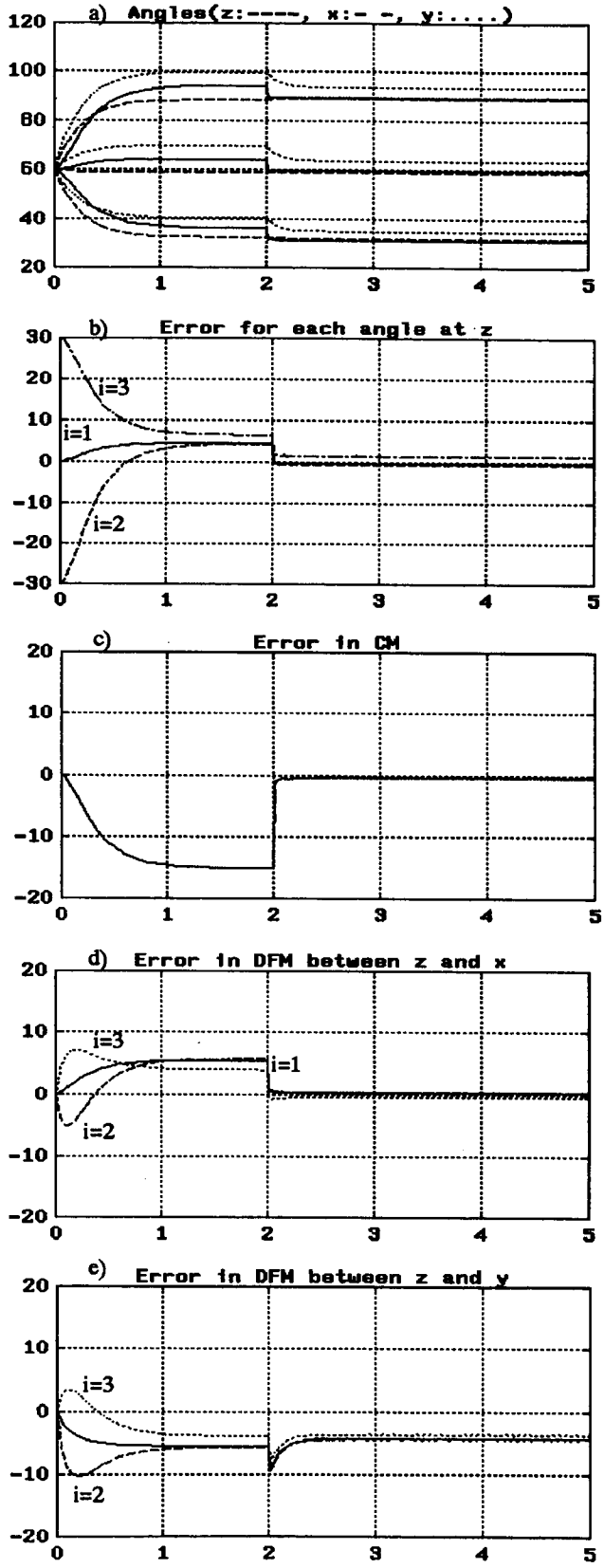


Fig. 7 Simulation results for the triangle in Fig. 5 with Backward Process started at t=2.

Actual Values	i=1	2	3
. Edge Normal, n_i:	60	150	300
. Angle, α_i:	60	90	30
Sensed Data			
. Edge Normal, sn_i:	59.5	151.0	298.5
. Angle, sa_i:	71.5	101.5	39.5
sb_i:	68.5	98.4	41.5
Variance of Sensor	angle:100,	edge-normal:25	
Time Constant:	$\tau_{xi}= \tau_{yi}= 4$,	$\tau_{zi}= 10$	

Tab.1 Actual and Sensed Data, and Sensor Parameters

in Fig. 7. Only with the forward process, **x** converges to equilibrium value which deviates from the real value considerably. However, it moves to another equilibrium value which is very close to the real value, just after the beginning of the backward process, as shown in a). The error between **x** and the real value is drawn in b) which shows clearly the error correcting effect of the backward process. The graphs in c), d) and e) also show the adjusting effects of the backward process on the errors in CM and DFMs.

CONCLUSION

This paper presents a theory of interactive and cooperative sensing and control as a fundamental paradigm of implementing advanced teleoperation. The proposed paradigm was intended to take full advantage of the current and future capabilities of a robot performing dextrous manipulation and sensor-based local autonomy.

A new method of achieving sensing-knowledge-command (SKC) fusion was presented as a basic computational mechanism for the proposed interactive and cooperative sensing and control.

A system architecture and man/machine interface protocol was described to show the preliminary implementation of the proposed system.

There still remains much work to do to refine and consolidate theory and implementation of the proposed interactive and cooperative sensing and control for advanced teleoperation.

REFERENCES

1. R. Goertz, "Some Work on Manipulator Systems at ANL, Past, Present, and a Look at the Future," PROC. 1964 SEMINARS ON REMOTELY OPERATED SPECIAL EQUIPMENT, U.S. Atomic Energy Commission Rept. CONF-640508.
2. R. Mosher, "Industrial Manipulators," SCIENTIFIC AMERICAN, vol. 211, pp. 88-96, Oct. 1964.
3. A. K. Bejczy and M. Handlykken, "Generalization of Bilateral Force-Reflecting Control of Manipulators," PROCEEDING OF 4TH RO-MAN-SYS, pp. 242-255, Warsaw, 1981.
4. S. Lee, "Interactive/Cooperative Automation Development for Robot Arms, Phase I : Generalized Bilateral Force Reflecting Control," USC ROBOTICS INSTITUTE REPORT RI 84-01, Los Angeles, CA, Mar. 1984.
5. S. Lee, G. Bekey and A.K. Bejczy, "Computer Control of Space-Born Teleoperators with Sensory Feedback," IEEE INT. CONF. ON ROBOTICS AND AUTOMATION, pp. 205-214, 1985.
6. A.K. Bejczy and J.K. Salisbury, "Kinematic Coupling between Operator and Remote Manipulator," INTERNATIONAL COMPUTER TECHNOLOGY CONFERENCE, *ASME*, Aug. 1980.
7. S. Hayati and S.T. Venkataraman, "Design and Implementation of a Robot Control System with Traded and Shared Control Capability," IEEE INT. CONF. ON ROBOTICS AND AUTOMATION, pp. 1310-1315, 1989.
8. Z. Szakaly, W.S. Kim and A.K. Bejczy, "Force-Reflecting Teleoperated System with Shared and Compliant Control Capabilities," PROC. NASA CONF. ON SPACE TELEROBOTICS, JPL Publication, Vol. 4, pp. 145-155, 1989.
9. J. Vertrut et al., "Advances in a Computer Aided Bi-

lateral Manipulator System," presented at THE ROBOTICS AND REMOTE HANDLING IN HOSTILE ENVIRONMENTS TOPICAL MEETING, Gatlinburg, 1984.

10. T. Sato, S. Hirai, and T. Horiba, "Language-Aided Master-Slave Manipulator System--Realization of Motion Constraint by Software Jig," IN PREPRINTS OF PROC. 25TH SICE CONF, pp. 359-360, 1982.

11. A.K. Bejczy, Z. Szakaly and W.S. Kim, "A Laboratory Breadboard System for Dual-Arm Teleoperation," SOAR' 89 WORKSHOP, pp. 649-660, NASA Johnson Space Center, Jul. 1989.

12. Ren C. Luo, Min-Hsiung Lin, and Ralph S. Scherp, "Dynamic Multi-Sensor Data Fusion System for Intelligent Robots," IEEE JOURNAL OF ROBOTICS AND AUTOMATION, Vol. 4, No. 4, pp. 386-385, August 1988.

13. Hans P. Moravec, "Sensor Fusion in Certainty Grids for Mobile Robots," AI MAGAZINE, pp. 61-74, Summer 1988.

14. Scott W. Shaw, Rui J. P. deFigueiredo, and Kumar Krishen, "Fusion of Radar and Optical Sensors For Space Robot Vision," IEEE INT. CONF. ON ROBOTICS AND AUTOMATION, pp. 1842-1846, 1988.

15. Tom Henderson and Esther Shilcrat, "Logical Sensor System," JOURNAL OF ROBOTIC SYSTEM, 1(2), pp. 169-193, 1984.

16. T. L. Huntsberger and S. N. Jayaramamurthy, "A Frame Work for Multi-Sensor Fusion in the Presence of Uncertainty," PROCEEDINGS OF SPATIAL REASONING AND MULTI-SENSOR FUSION WORKSHOP, pp. 345-350, 1987.

17. Hugh F. Durrant-Whyte, "Sensor Models and Multi-Sensor Integration," IEEE JOURNAL OF ROBOTICS AND AUTOMATION, 1987.

18. Sukhan Lee, Paul S. Schenker, and Jun Park, "Sensor-Knowledge-Command Fusion Paradigm for Man/Machine Systems," PROCEEDINS OF SPIE'S INTERNATIONAL SYMPOSIUM ON ADVANCED INTELLIGENT SYSTEMS, Nov. 1990, Boston, MA.

N93-32109

AIR FORCE RESEARCH IN HUMAN SENSORY FEEDBACK FOR TELEPRESENCE

Ronald G. Julian
Captain, USAF
Armstrong Laboratory AL/CFBA
Wright-Patterson AFB OH 45433-6573

Abstract

Telepresence operations require high quality information transfer between the human master and the remotely located slave. Present Air Force research focusses on the human aspects of the information needed to complete the control/feedback loop. Work in three key areas of human sensory feedback for manipulation of objects are described. Specific projects in each key area are outlined, including research tools (hardware), planned research, and test results. Non-manipulative feedback technologies are mentioned to complete the advanced teleoperation discussions.

Introduction

The Air Force renewed its interest in remotely operated manipulator systems in 1985 with Project Forecast II. The current concept is based on the idea of a high quality teleoperated master-slave manipulator system [1]-[2]. The present program focus is centered on the feedback signals that must be generated to adequately display force information to the human operator.

The key to the successful fielding of teleoperator systems is the design and implementation of a good human-machine interface. Three human senses (touching, hearing, and seeing) hold the highest potential for effective human sensory feedback. While each sense is somewhat independent, the senses also complement each other in the human though the interdependence is not well understood. Humans use visual feedback to gather large amounts of high frequency, high resolution information about the environment. Likewise, a telepresence system can present significant quantities of information via visual displays. However, the visual system is ineffective in some environments such as heavy smoke, darkness or situations with obstructed vision. In these cases, humans often compensate with audio or tactile feedback. Audio feedback can provide information that is essentially omnidirectional and invisible. Touch and proprioceptive feedback can provide critical information regarding manipulative tasks. Tactile feedback can supplement visual feedback and in many cases permit completion of the task without vision.

Remote Manipulation Spectrum

Prior to discussing the details of the human sensory feedback (HSF), it is important to identify the areas within the teleoperation spectrum where HSF is important. Sheridan [3] offers an excellent review of the subject and provides well-ordered definitions to each key area. Sheridan's definitions are included to provide a common point of departure:

"*Telepresence* is the ideal of sensing sufficient information about the teleoperator and task environment, and communicating this to the human operator in a sufficiently natural way, that the operator feels physically at the remote site."

"*Teleoperation* is the extension of a person's sensing and manipulation capability to a remote location. A teleoperator includes at the minimum artificial sensors, arms and hands, a vehicle for carrying these, and communication channels to and from the human operator. The term "teleoperation" refers most commonly to direct and continuous human control of the teleoperator, but can also be used generally to encompass "telerobotics" (see below) as well."

"*Telerobotics* is a form of teleoperation in which a human operator

acts as a supervisor, intermittently communicating to a computer about goals, constraints, plans, contingencies, assumptions, suggestions, and orders relative to a limited task, getting back information about accomplishments, difficulties, concerns, and, as requested, raw sensory data--while the subordinate telerobot executes the task based on information received from the human operator plus its own artificial sensing and intelligence."

"*Supervisory Control* in the present context is mostly synonymous with telerobotics, referring to the analogy of a human supervisor directing and monitoring the activities of a human subordinate. The term "supervisory control" is used commonly to refer to human supervision of any semi-autonomous system (including an aircraft, a chemical or power plant, etc.), while 'telerobot' commonly refers to a device having arms for manipulating or processing discrete objects in its environment."

"*Robotics* is the science and art of performing, by means of an automatic apparatus or device, functions ordinarily ascribed to human beings, or operating with what appears to be almost human intelligence (adapted from *Webster's Third International Dictionary*)."

As can be seen in Figure 1, the level of human operator involvement varies with each key area.

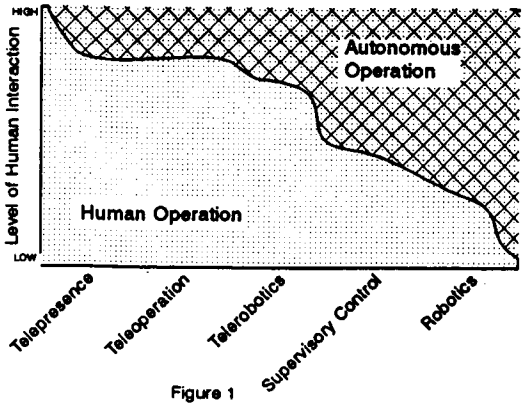


Figure 1

Armstrong Laboratory Research

Human Sensory Feedback Research at the Armstrong Laboratory's Crew Systems directorate is centered in three areas of the force feedback domain:

Coarse Positioning - Movement produced by large scale motion associated with the human arm. Coarse manipulation implies movement to position a dexterous end effector in the proper orientation to perform work via fine manipulation.

Fine Manipulation - Small scale motion associated with the human hand. Fine manipulation is the performance of tasks using highly dexterous end effectors.

Tactile Feedback - Determine the role of tactile feedback from the remote system to the human operator. Identify and investigate technologies important to providing high fidelity tactile feedback to human operators.

Coarse Positioning Research

Current coarse positioning investigations are evaluating the impact of exoskeletal systems on human performance. The method is to measure human performance using an instrumented "peg-in-the-hole' task board based on Fitts' Law [4][5]. The unencumbered operator executes a test sequence to obtain a baseline of task performance. The operator then dons a non-force-reflecting (NFR) exoskeleton and re-executes the test sequence. The difference yields an indicator of the restriction of the exoskeleton on the human's task performance. Results of the initial study show a 32 to 41 percent reduction in task performance. This reduction in task performance is due solely to the interference caused by the NFR exoskeleton.

Follow-on testing will evaluate task performance as the human operator uses the exoskeleton to teleoperate a slave robot. The result will be a system-dependent indicator of the impact of the total master-slave system on the task performance. The preliminary testing was done with a NFR exoskeleton. The final sequence in this phase will completed with a NFR exoskeleton. This NFR exoskele-

ton encumbrance test forms the basis for subsequent testing using the force reflecting exoskeleton described below.

For the past two years, Odetics, Inc., has been developing a Force REFLecting EXoskeleton (FREFLEX) to evaluate the impact on task performance of force reflection with a non-constraining seven-degree-of-freedom force-reflecting exoskeleton. Design of the cable-driven system is discussed in [6].

A similar series of test sequences will be conducted using the FREFLEX. The encumbrance of the FREFLEX will be measured in the NFR mode as before. Once the passive baseline is determined, a series of force reflecting experiments will be conducted to evaluate several parameters such as gravity and friction compensation, feedforward and feedback control algorithms, and modified kinematic and dynamic parameters. The expected result is an increased understanding of each of the factors that impact the human operator in a complex telepresence system. The end goal is to develop design parameter guidelines that are based on knowledge of the human characteristics in the telepresence system.

A second interest area in coarse positioning is to quantify kinematic and dynamic limitations placed on the operator by the exoskeleton system. Research in this area is centered on the WATSMART™[14], a three dimensional motion analysis system. The approach is to measure the operator's kinematic and dynamic parameters with and without the exoskeleton. The objective is to examine the joint-by-joint kinematic and dynamic encumbrances placed on the operator by the exoskeleton. The resultant data is expected to lead to human-based parametric design guidelines for exoskeletons.

The WATSMART™ data will also be used to establish velocity and acceleration profiles of each joint. In addition to evaluating the encumbrance effects, these velocity and acceleration profiles are necessary to develop mechanical performance parameters of motors and actuators for future exoskeleton devices.

Fine Manipulation Research

Fine manipulation research at the Armstrong Laboratory is centered on the pair of dexterous hands designed by Jacobson at the University of Utah [7]. Initial plans were to combine the robotic hands with the robotic arms to form a slave system for telepresence research. Funding reductions have prevented arm-hand integration but theoretical research has continued. Whalen [8] developed a basis for planar grasping as a foundation for dexterous robotic hands. Future work will be directed toward understanding the relationship of each digit of the robotic hand.

A key component to successful fine manipulation via telepresence is the development of kinematic and dynamic models between the slave hand, the operator's hand-master controller, and the operator's hand. Initial kinematic mapping from the slave hand to the master hand controller to the human operator's hand has been completed [9]. Force feedback for hand-master controllers will require innovative actuation schemes. Initial thoughts on hand master controllers with feedback indicate requirements for high speed, lightweight actuators with high power densities and approximately linear responses. Current plans call for feedback signals for the acutators to be derived from tendon tension signals on the dexterous hands. Long term objectives will combine the hand master control loop with the slave hand control loop to produce a force feedback capability in fine manipulation. Auxiliary force information from tactile sensors on the hands may also be integrated into the overall feedback scheme to provide even more stable control.

Tactile Feedback Research

The third, but much less understood, area of force feedback is tactile feedback. Tactile feedback plays a critical role in human performance. Armstrong Laboratory work in this area has two key thrusts: (1) The addition of sensors to one of the dexterous hands (2) the development of small lightweight tactile stimulators.

A Phase II Small Business Innovation

Research (SBIR) project will yield a sensorized robotic hand in FY93. The sensor suite includes palmar sensors as well as sensors on the back of the robotic fingers. Additional sensor research is ongoing at the Air Force Institute of Technology. Nering [10] is developing a slip sensor using resistive paint and artificial neural networks. Other AFIT researchers [11][12] have explored the use of polyvinlylidene fluoride film to fabricate high-density contact position and force sensors on silicon substrate.

The second component of the tactile feedback subsystem is the tactile stimulator for the human operator's hand. While a specific technology has not been selected, present work focusses on Shape Memory Alloy (SMA). The current device is a five-by-six (30-element) array using SMA actuators for stimulator pins[13].

Psychophysical perception testing on human subjects is planned to evaluate the SMA applicability to tactile stimulator technology. Follow-on investigations will explore operator capability to detect active and static patterns and to identify dynamic patterns. Long term plans call for the integration of the robotic tactile sensors with the operator tactile stimulators to achieve a closed loop feedback subsystem in the tactile domain.

Conclusion:

Armstrong Laboratory research in human-in-the-loop controlled robots is focusing in three key areas. Coarse positioning, using force-reflecting exoskeletons, is the key element in positioning the robotic end effector to do work. Once the coarse positioning subsystem has the manipulator in place, the fine manipulation task will be performed by highly dexterous robotic hands. Force feedback to the human operator via a force-reflecting hand master is planned. Tactile feedback research centers on shape memory alloy technology for tactile stimulators. The stimulators will be combined with the robotic tactile sensor suites to yield a complete tactile subsystem.

Long term plans call for integration of the three manipulation areas into a single manipulation system with intuitive force feedback. It is anticipated that video feedback subsystems and audio subsystems will have off-the-shelf availability when the manipulation matures sufficiently for total system integration.

References:

[1] Sheridan, T.B., Ferrell, W.R., "Supervisory Control of Manipulation," In Third Annual NASA-University Conference on Manual Control, NASA SP-144, National Aeronautics and Space Administration, Washington DC, 1967.

[2] Julian, R.G., Anderson, T.R., "Robotic Telepresence: Applications of Human-Controlled Robots in Air Force Maintenance," Aerospace Simulation 1988: SCS Multi-conference on Aerospace Simulation III, pp59-67, 1988.

[3] Sheridan, T.B., "Telerobotics", Automatica, vol 25, no 4, pp 487-507, 1989 International Federation of Automatic Control, 1989.

[4] Fitts, P.M., "The Information Capacity of the Human Motor Control System in Controlling the Amplitude of a Movement," Journal of Experimental Psychology, vol 47, pp 381-391, 1954.

[5] Remis, S.J., Repperger, D.W., "Quantifying Performance Degradation Due to the Human-Machine Interface of Telemanipulators," AAMRL-SR-90-510, Harry G. Armstrong Aerospace Medical Research Laboratory, 1990.

[6] Burke, J.B., Nelson, D.K., "Exoskeleton Master Controller with Force-Reflecting Telepresence," NASA Conference Publication 3127, Proceedings of 1991 Space Operations, Applications, and Research Symposium, pp 321-330, 1991.

[7] Jacobson, S.C., Wood, J.E., Knutti, D.F., Biggers, K.B., "The UTAH/MIT Dexterous Hand: Work in Progress," Int. J. of Robotics Research, vol 3, no 4, pp 21-50, 1984.

[8] Whalen, P.V., "Two-Fingered Grasps of Cylindrical Objects in Planar Motion," AFIT/GAE/AA/88D-39, Air Force Institute of Technology,

1988.

[9] Wright, A.K., Stanisic, M.M., "Kinematic Mapping Between the EXOS Handmaster Exoskeleton and the Utah/MIT Dexterous Hand, Proceedings of the IEEE International Conference on Systems Engineering, pp 101-104, 1990.

[10] Nering, J., Masters thesis to be published, Air Force Institute of Technology, 1993

[11] Reston, R., "Robotic Tactile Sensor Fabrication from Piezoelectric Polyvinylidene Fluoride Films," AFIT-/GE/ENG/88D-41, Air Force Institute of Technology, 1988

[12] Ford, D., "Multiplexed Robotic Tactile Sensor Fabricated from Polyvinylidene Fluoride Film," AFIT/GE/-ENG/89D-13, Air Force Institute of Technology, 1989

[13] Johnson, A.D., "Shape Memory Alloy Tactile Feedback Actuator," AAMRL-TR-90-39, Harry G. Armstrong Aerospace Medical Research Laboratory, 1990.

[14] Operator's Manual, Northern Digital Inc, Waterloo, Ontario, Canada, 1989

Session R4: TERRESTRIAL/UNDERWATER ROBOTIC SYSTEMS

Session Chair: Tom Davis

N93-32110

Air Force Construction Automation/Robotics

A.D. Nease, E.F. Alexander
Air Force Civil Engineering Support Agency
Air Base Operability and Repair Branch
Tyndall AFB, Fl 32403

ABSTRACT

The Air Force has several missions which generate unique requirements that are being met through the development of construction robotic technology. One especially important mission will be the conduct of Department of Defense (DOD) space activities. Space operations and other missions place construction/repair equipment operators in dangerous environments and potentially harmful situations. Additionally, force reductions require that human resources be leveraged to the maximum extent possible and more stringent construction repair requirements push for increased automation. To solve these problems, the US Air Force is undertaking a research and development effort at Tyndall AFB, Fl., to develop robotic construction/repair equipment. This development effort involves the following technologies: teleoperation, telerobotics, construction operations (excavation, grading, leveling, tool change), robotic vehicle communications, vehicle navigation, mission/vehicle task control architecture and associated computing environment. The ultimate goal is the fielding of a robotic repair capability operating at the level of supervised autonomy. This paper will discuss current and planned efforts in space construction/repair, explosive ordnance disposal, hazardous waste cleanup, and fire fighting.

INTRODUCTION

The policy of the Air Force (AF) with regard to space is that, given the mission, structure, expertise, and history of the AF, it is the service especially and uniquely qualified to carry out DOD space operations. Additionally, the AF will be the major provider of space forces for the nation's defense. The Air Force envisions three areas of space operations: earth-based/launch support, orbital, and lunar/extra-terrestrial.

The work described in this paper is being done at the AF Civil Engineering Laboratory (CEL) which supports the AF Civil Engineering and Services (CE&S) organization. The CE&S Space Master Plan, in the time frame present-to-2000, specifically requires that advances in robotics technology be sought to enhance ongoing operations in dangerous environments. In the time frame 2000-and-beyond, the plan calls for the determination and development of uses of robotic construction equipment developed for terrestrial AF civil engineering work to be applied to extra-terrestrial construction operations.

The current robotic technology development effort at the CEL is sponsored by the Office of Secretary of Defense (OSD) Unmanned Ground Vehicle (UGV) program. This project will develop a telerobotic means of executing post-attack (and peacetime) Explosive Ordnance Disposal (EOD) and operating surface repair and recovery. As manpower becomes more critical to the Air Force of the future, the benefits of mobile robotics (UGV's) to the wartime recovery mission and peacetime range clearance missions become more obvious and important. The UGV can perform the mission in the post-attack environment while human operators and other airbase personnel remain in a safe shelter.

It should be noted that there are other missions, in addition to space construction and airbase recovery, under the purview of the Air Force Civil Engineer that lend themselves to application of the technology being developed under the OSD/UGV RRR program. As mentioned above, weapons range clearance of UXO is a good use of robotics technology. Airbase fire fighting/fire detection and hazardous waste handling/cleanup are other areas that will benefit from this technology. (see Figure 1)(PIE CHART)

CRITICAL TECHNOLOGIES

The Air Force Civil Engineering Support Agency (AFCESA) has developed a telerobotic excavator, a John Deere 690C, that has a teleoperational capability as well as on-board smarts in the form of preprogrammed functions. Drawing on technological and operational concepts generated, in part, from testing the John Deere telerobotic excavator system, efforts toward producing a mature RRR/Construction UGV concentrate on several critical technology areas: a) communications, b) navigation/guidance, c) mapping/sensors, d) vehicle/platform, and e) task control architecture/computing environment.

a. Communications:

The RRR/construction automation communications system is being developed in two phases with a demonstration planned for each phase. In phase-1, the fixed operator control station and the existing RRR robotics excavator communications' links will be developed and tested to evaluate their performance in a PC environment with a single vehicle. Phase-2 will involve multiple vehicle communications in a VME-based environment. The command and control link will provide two-way digital data link between the control center and the vehicle(s). This link will operate at 9600 baud at ranges up to 5 miles in the 1433 MHz band. Responses to the control center over the return link will occur only from the specific vehicle addressed by a preceding control center transmission over the C/C link (Fig 2). A one-way video link will transmit a standard 525 line color or B/W signal at 1795.5 MHz. In addition to the above, the comm system will format and encode data for transmissions over the identified system links, decode

received transmissions, manage link operation, and interface the links to the end using equipment.

b. Navigation:

A robotic construction vehicle navigation system is being developed to provide the capability for the repair vehicle to move autonomously from its storage area/shelter to the repair site and then to move autonomously among several repair sites and return to the storage area upon completion of the mission. The heart of the system is a Modular Azimuth Positioning System (MAPS) ring laser gyro inertial navigation system. The MAPS will be updated at approximately 3 minute intervals by a kinematic differential Global Positioning System (GPS). The kinematic differential GPS builds on accuracy of standard differential GPS by comparison of phase differences between reference station and remote receiver. The nav system will be given initial vehicle position/orientation, repair site location (digital map), and first repair site position and required orientation. A plan for vehicle motion will be generated off-line and executed with the aid of obstacle detection sensors. (Fig 3) The nav system is being developed on a "mule", a small 4-wheel vehicle used to provide an "anywhere" test capability. This system will be demonstrated in early 1993.

c: Mapping/Sensors:

The objective of the mapping/sensor development effort is to provide an autonomous capability to characterize the crater/repair site. It is given that the vehicle reference point remains constant and that displacement is known from the reference. First, a 2-dimensional camera and an electronic distance measurement (EDM) device will be used to size the crater/repair site from a stationary vehicle. Multiple images will be spliced together to form the composite 2-D image. Image analysis of the collected data of the crater/repair site will be used to estimate the location of the perimeter and the deepest part of the site. This information will be verified using EDM to locate and confirm the actual perimeter versus false edges created by lighting problems. A 3-D wire frame, perspective view, model will be laid over the 2-D image with vertices at all confirmed points. The volume of the crater/repair site will be estimated using first and second order approximations of the wire frame model. The multiple 3-D images will be spliced together to form the work space environment containing as much of the repair site area as possible. The information displayed will be the classification and centroids of objects of interest in the area as well as the edge of the repair site. The data will be provided to the vehicle control system. Analysis and tests will be made to incorporate information from the second ambiguity region of the scanner. Rules for use of the laser scanner will be developed where appropriate as well as scene requirements and conditions. The initial demo will be done from a fixed base and will show the capability of this technology and the

associated software to solve the problems involved with mapping/characterizing a repair site for a remote operator and/or an autonomous vehicle.

d. Testbed Vehicle:

The new robotic testbed vehicle will be designed and built from the ground up as a next-generation construction robot (Fig 4). This vehicle will be an all-terrain, low ground pressure, rubber tracked machine powered by a 250 hp Caterpillar diesel engine. A 5kw diesel generator that can provide AC and DC power for the on-board electronics as well as for field testing of systems will be located forward of the main diesel engine. The vehicle hydraulic system will be a closed-center, load-compensating system that will feature in-cylinder position/force sensors and servo valves mounted directly on the cylinders/actuators. The VME-based, on-board computer will be housed in a shock-proof, climate controlled enclosure designed for easy access. The key to this vehicle is that it will build on the lessons learned with the existing robotic excavator.

e. Architecture and Computing Environment:

The system development platform consists of Sun Sparc/2 workstations which provide a UNIX platform for compilation and development. Graphics/simulation work will be done on a Silicon Graphics IRIS 4D/310VGX workstation. All units are interconnected via Ethernet communications links (Fig 5). The VXWorks Operating System provides a UNIX-like realtime multitasking platform for VME targets. The OS allows for a small, re configurable kernel and provides the necessary speed and multitasking capabilities. VXWorks enjoys wide R&D community support as do the Sun and Silicon Graphics platforms.

The control station and remote vehicle platforms act as VME targets that allow for multiple CPU processing and exchange of information across a common bus. The multiple CPU's will process concurrently while individual CPU's process multiple tasks concurrently. The individual CPU's are dedicated to functional areas or tasks and exchange information with the common memory area.

The control station functional structure will provide a communications module that communicates between remote vehicles and the control station. This link also provides for communications between CPU's across the bus. This system provides separate channels for emergency messages, remote control functions, and vehicle status information. The operator interface module provides functions for remote control, displays status information, provides the interface to autonomous functions and routines for emergency or error handling. The task coordination module provides coordination between the remote vehicles. The damage assessment module incorporates damage information into a global repair area map.

The remote vehicle functional structure consists of a system planning and arbitration module that performs overall task planning, task control, error handling, and arbitration of system resources for contending CPU's and tasks. The navigation & guidance module integrates INS, GPS, and embedded vehicle sensor information and performs path planning. This module also integrates IR/acoustic sensor information and performs rudimentary obstacle avoidance. The vehicle control module integrates sensor information to initiate and control vehicle movement. The imaging module integrates laser scanning and video images to produce a local repair map for the system planning module. The sensor management module samples all sensors and updates common memory.

CONCLUSION

Much of the technology being developed under the current CEL construction automation program can be used as a base for robotic space construction activities. Because of the expense of supporting humans in the lunar environment, very few will be available for the actual construction and maintenance of facilities. Semi-autonomous operation will be the rule rather than the exception for lunar construction machinery.

Some special considerations for lunar construction equipment will be basic design problems. Machine mass is obviously critical. Boosting mass from the earth to the lunar surface is incredibly expensive. The upper affordable limit for a lunar construction robot is probably 40,000 lb. Since traction is a linear function of vehicle mass, operation in 1/6 g on the moon will mean using innovative anchoring techniques or adding mass once on the moon. The addition of mass to the vehicle brings up a whole new set of problems: manueverability, inertia, etc. Sealing and lubrication of moving parts in a hard vacuum and in the presence of the lunar dust will be a challenge. A key development in the AF construction robotics program has been that of the multipurpose machine with the capability to carry and change, as required, an entire suite of tools. It's obvious that this will be an important attribute of the lunar/extra-terrestrial construction robot. Clearly, none of these problems is insurmountable, and the time to begin solving them is now.

System integration for the current OSD-funded program will be conducted at Tyndall Air Force Base as an in-house effort. The AFCESA/CEL robotics program currently has 8 engineers and support personnel and a laboratory dedicated to the successful completion of the program. The Air Force realizes the critical nature of robotics development for missions that are hazardous, repetitive or both. The array of missions to which this technology can be applied is very broad and certainly includes space construction (lunar/extra-terrestrial) and earth-based (launch support). The bottom line requirement for robotic construction equipment can be summed up in four words: FASTER, SAFER, CHEAPER, BETTER.

CONSTRUCTION ROBOTICS

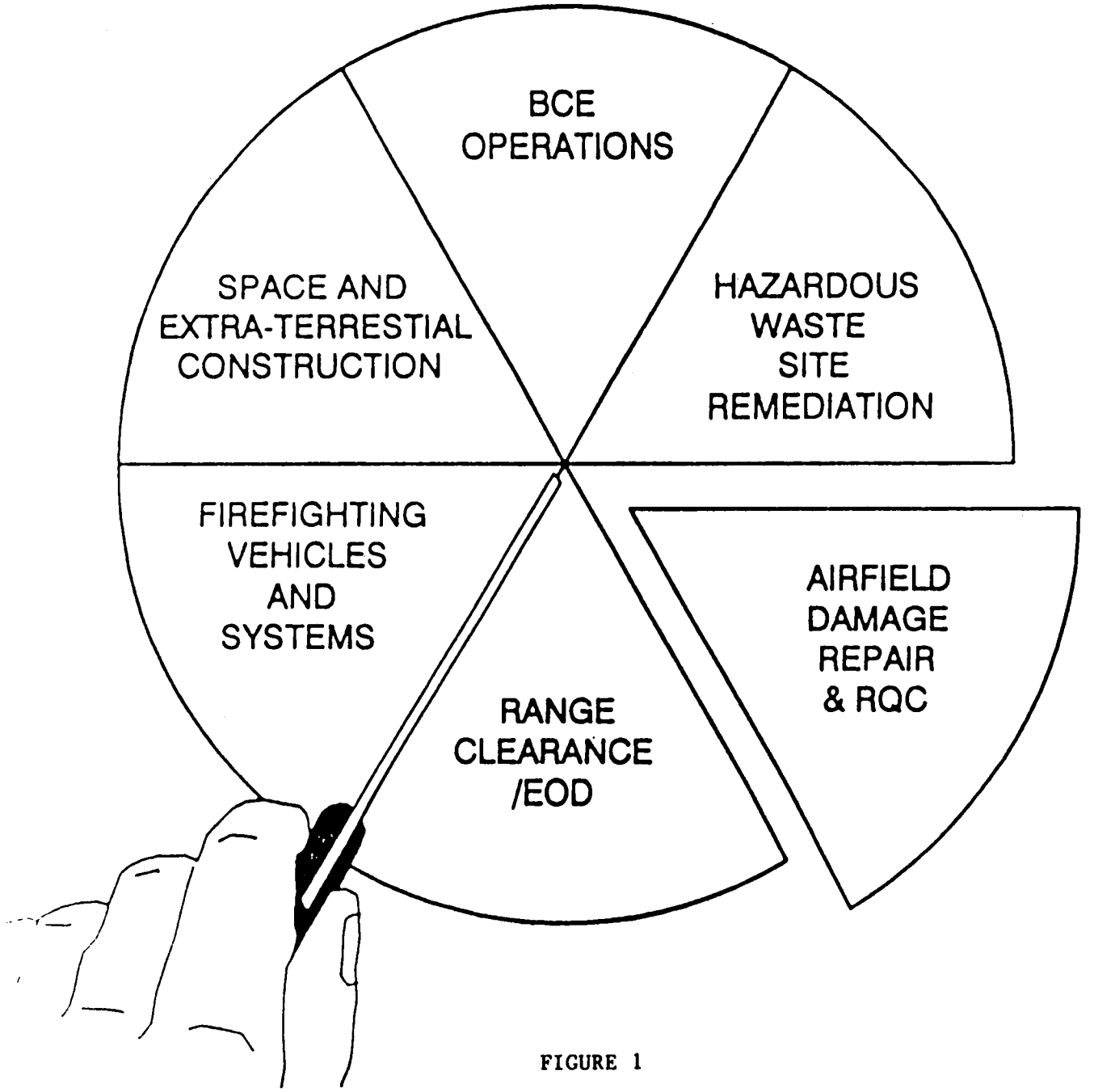


FIGURE 1

Deomonstration System Baseline Configuration

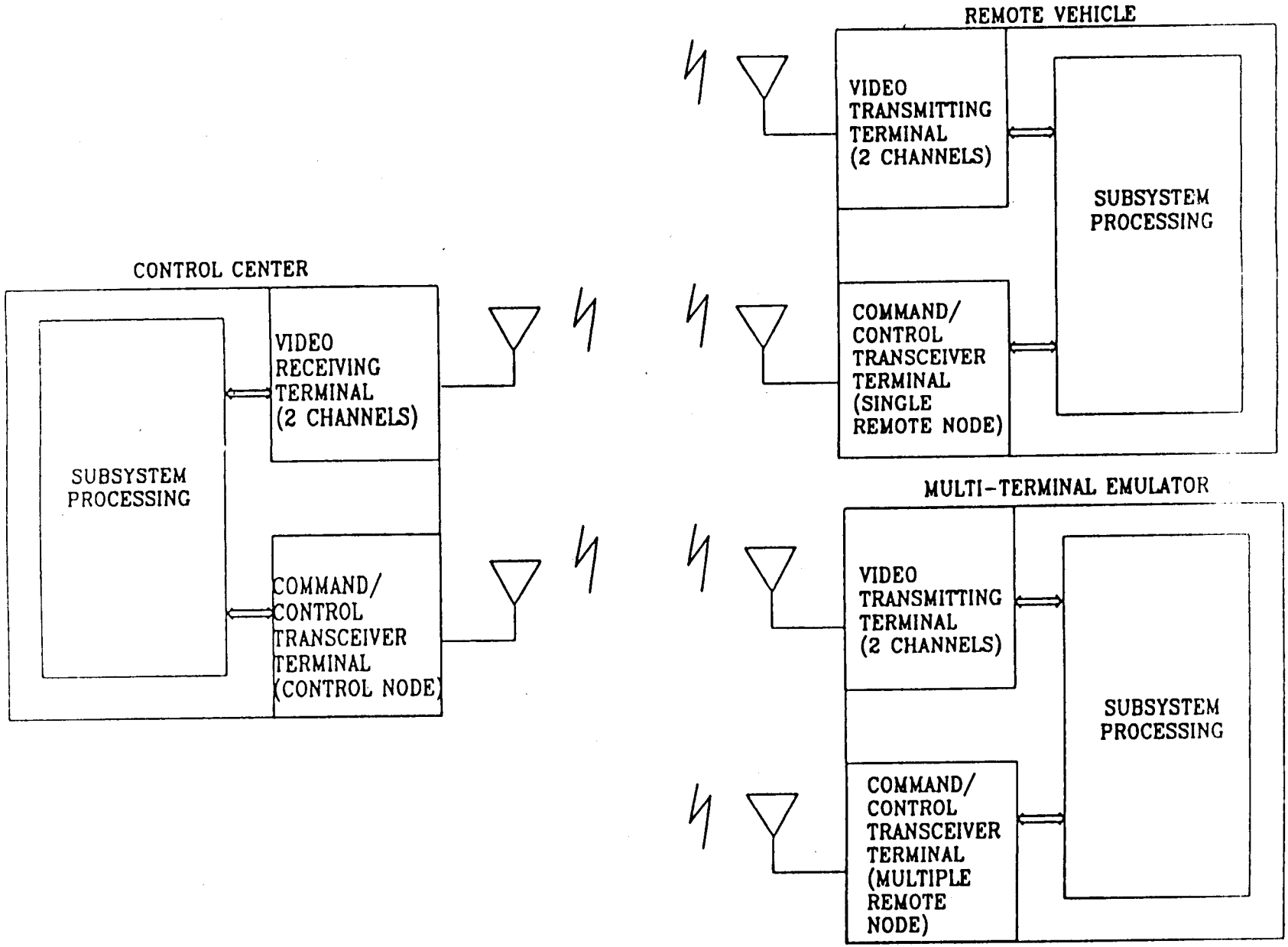


FIGURE 2

Off-Line Path Planning

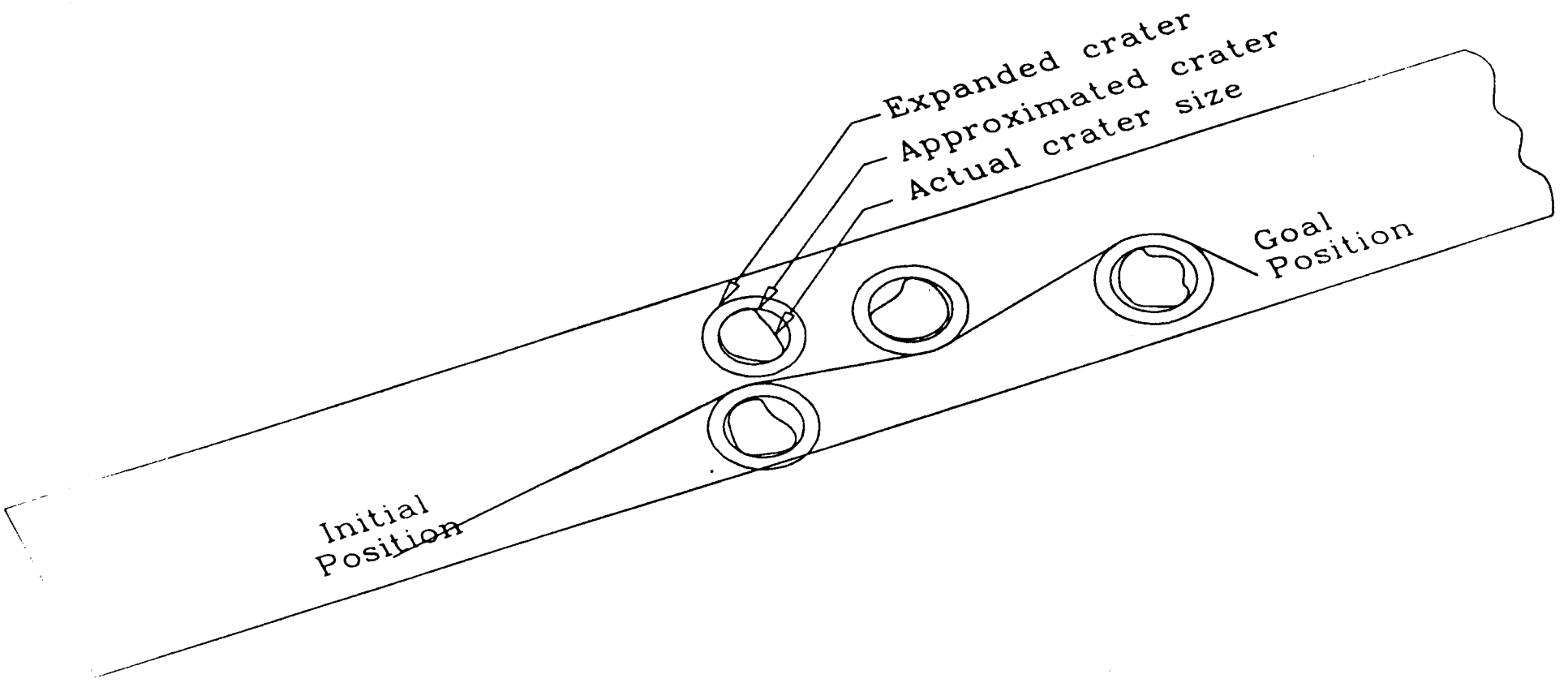


FIGURE 3

CONSTRUCTION ROBOT

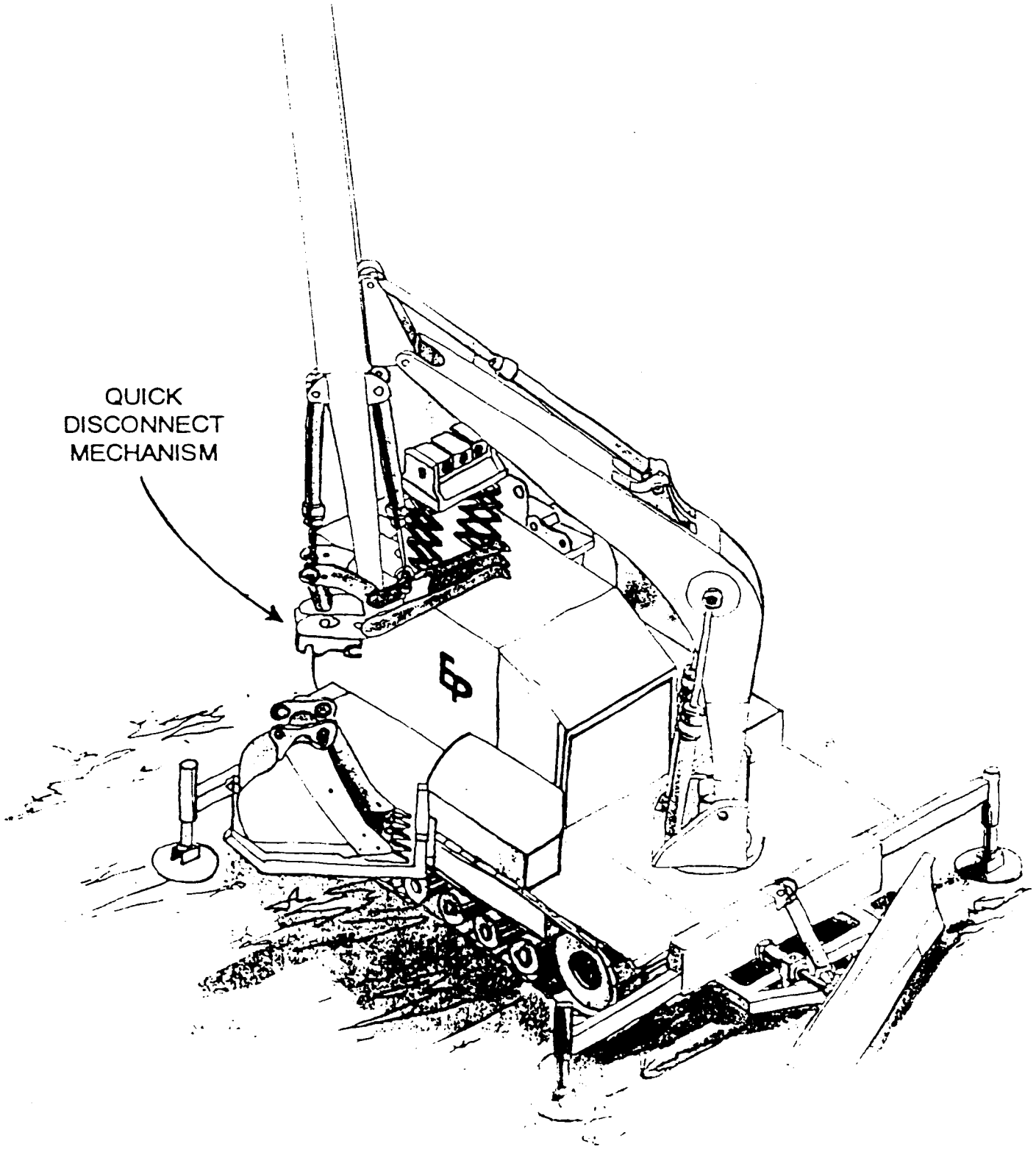


FIGURE 4

Intelligent Control Hardware Architecture

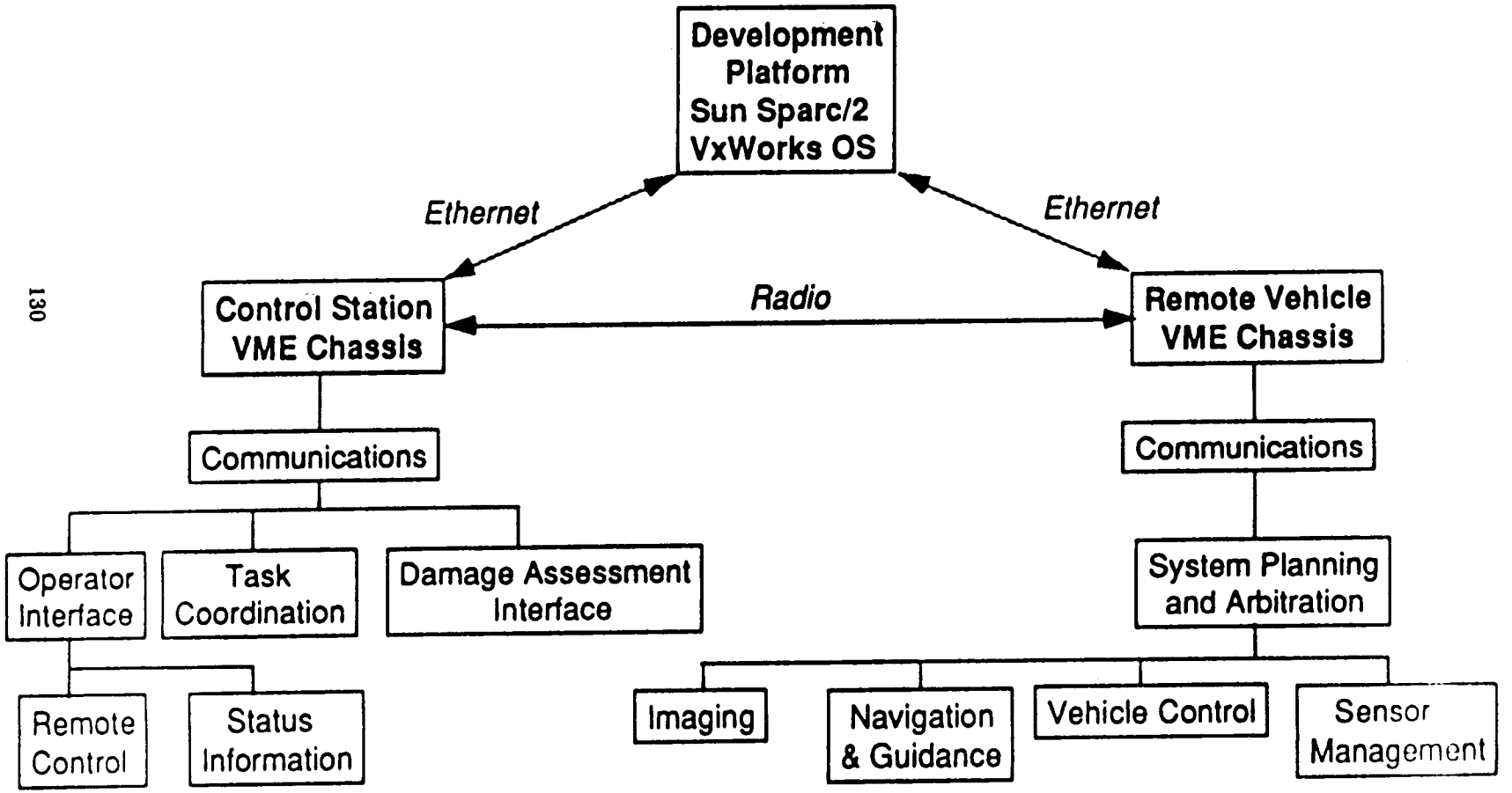


FIGURE 5

SATELLITE TEST ASSISTANT ROBOT (STAR)[1]

D. A. McAffee[2]
D. J. Kerrisk[3]
K. R. Johnson[3]

Jet Propulsion Laboratory
Pasadena, California

ABSTRACT

A three-year, three-phase program to demonstrate the applicability of telerobotic technology to the testing of satellites and other spacecraft has been initiated. Specifically, the objectives are to design, fabricate, and install into the JPL 25-ft. Space Simulator (SS) a system that will provide the capability to view test articles from all directions in both the visible and infrared (IR) spectral regions, to automatically map the solar flux intensity over the entire work volume of the chamber, and to provide the capability for leak detection.

The first year's work, which provides a vertically mobile viewing platform equipped with stereo cameras, will be discussed. Design constraints and system implementation approaches mandated by the requirements of thermal vacuum operation will be emphasized.

INTRODUCTION

Telerobotics in the general domain of space applications has had a difficult time in attracting the support of a user community. This is not surprising; flight system managers tend to be very conservative technologically, and rightly so. No flight system manager is likely to be willing to put at hazard his budget and schedule in order to incorporate into his program new and unproven technologies that are not essential to his primary mission objectives.

To break this impasse, it will be necessary for spacecraft program managers to see the capabilities of telerobotics in action, and to be able to judge the maturity of the technology, in a non-threatening environment. One such environment that could have high visibility to spacecraft managers, but still be non-threatening, is spacecraft testing.

---

1 This paper presents one aspect of the work carried out by the Jet Propulsion Laboratory for the National Aeronautics and Space Administration

2 Member of the Technical Staff

3 Technical Group Supervisor

The STAR program was devised to fill an observed need (for a greater degree of automation and more flexibility) in the spacecraft testing arena. It was proposed to Code R as a joint effort between the telerobotics technologists and those responsible for spacecraft testing, as a means for introducing technology developed under the aegis of the Telerobotics Program into a flight program environment in a manner that would be non-threatening to flight articles. At the same time it presented to the Code R program the challenge of designing to an environment close to that of space. The emphasis in the program therefore is on the detailed engineering required to adapt known technology to the harsh environment of space. It is the flavor of that detailed engineering that this paper attempts to convey.

GROUND RULES

The two ground rules that guided the conception of the program were:

(1) the effort should showcase technology developed by Code R as part of the Telerobotics Program; and

(2) the product should be sufficiently attractive and non-threatening to the user community that they would be willing to incorporate it into their test plans.

To satisfy these ground rules a joint effort was initiated between the telerobotics technologists and the spacecraft test engineers to identify those needs that might best be met by telerobotics technology. An obvious target was test operations in the large space simulators in which spacecraft system testing is conducted. Here, where the test article is inaccessible during test, and where gaining access to the test article is both expensive and time-consuming, it seemed that telerobotic techniques could prove valuable in assisting the test operations. A crude estimate indicated test cost savings of a quarter of a million dollars per year might be anticipated. One major constraint was immediately recognized, however: nothing (and certainly no robotic element) would be allowed to penetrate the work volume of any spacecraft while it was in the chamber. Thus, any assistance during test operations would be limited to remote sensing. Even with this constraint, however, there were immediately identified a number of functions that telerobotics-developed technology might supply.

Remote observation- direct observation of test articles is extremely limited. In order to provide a uniform thermal background, the number and size of observation ports in the two major JPL test chambers has been kept to a minimum, and the viewing angles available are far from ideal. This has been somewhat compensated in recent years by cameras mounted in the chamber, but these have been in fixed locations. The ability to observe the test article from all angles, at varying degrees of

magnification, and in stereo, was identified as a highly desirable capability to have.

Remote temperature sensing- presently spacecraft are instrumented with hundreds of thermocouples to provide verification of thermal models. If an IR camera could be mounted to provide viewing of the test article from any and all angles, a great deal more data could be generated with a great deal less effort. IR sensing of the test article from the movable pan/tilt platform was also identified as a desirable capability.

Solar intensity mapping- obtaining a map of solar intensity throughout the chamber is a cumbersome and laborious process as presently implemented. Automation of this measurement, which is carried out generally only when no test article is present, would be another desirable feature for any system to be installed in the chamber.

Leak detection- pinpointing leaks in the shroud when they occur is a very difficult and time-consuming task, involving going over the surface with helium leak detectors. Any automation of this function, which would result in reduction of chamber down-time, would be very useful. It was estimated that this alone could save $120,000 per year.

Having identified a list of potential functions, a design concept for a system to provide those functions was generated, and is shown in Figure 1. A program was then outlined that would allow a phased development of capability, with checkpoints along the way that would allow periodic reevaluation of both objectives and progress. Providing users with an early demonstration of the potential advantages of the technology was an important aspect of the program, which was proposed in three phases:

Phase 1- FY '92- Demonstrate in the JPL 10-ft SS an improved viewing capability with a Z-axis-movable pan/tilt platform on which a stereo vision system is mounted. (Because of previously scheduled modifications, the 25-ft SS will not be available for STAR installation until the end of FY 93.)

Phase 2- FY '93- Install the system into the 25-ft SS, and add the IR camera and the solar spot mapping capability.

Phase 3- FY '94- Add capability for azimuthal motion of the platform, and leak detection capbility.

This paper presents the Phase 1 effort.

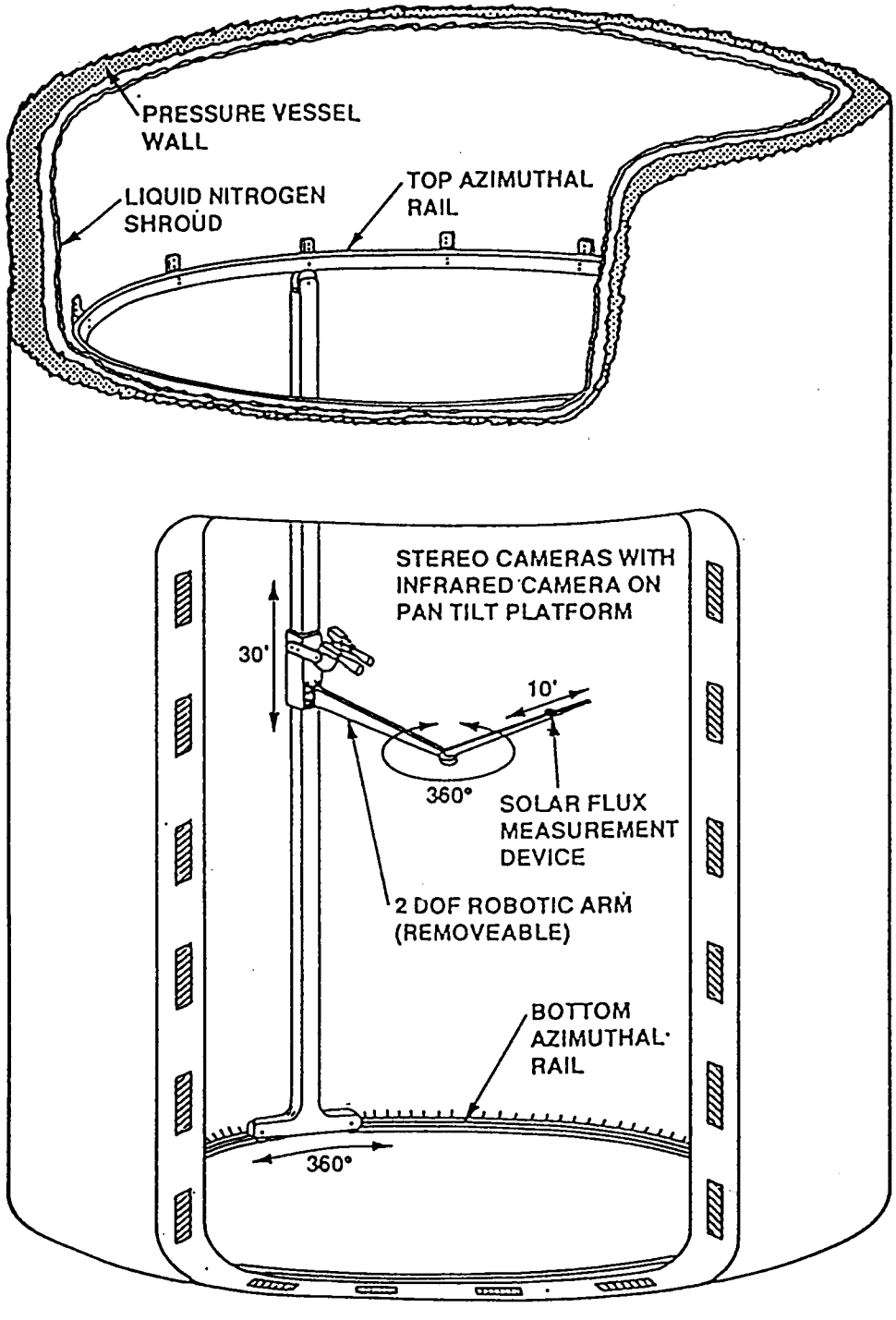


Figure 1. Initial Concept for the STAR System

MAJOR DESIGN CONSIDERATIONS

The basic concept for the system as outlined above is quite straightforward. Complications quickly arise however when specific requirements related to the application are taken into consideration. First and foremost of these is the environment in which the system will have to operate.

For STAR, the application environment includes high vacuum, i.e., on the order of 10^{-7} torr, and temperatures ranging from -196° C to +93°C. (It should be noted here that for the Phase 1 demonstration there will be no cold shroud in the 10-ft SS. However, since the same hardware is to be ultimately installed in the 25-ft SS, the design temperature range must accommodate that application.) Hard vacuum operation imposes stringent cleanliness requirements, since no significant outgassing can be tolerated, both in terms of maintaining vacuum, and in avoiding contamination of the test article. Further, rubbing surfaces are to be avoided, since they tend to produce particulates which can then deposit on sensitive surfaces. The large temperature range means that there will be significant dimensional changes in all components; these changes will of course be a function of the materials used. The design must address these considerations in detail.

Another important design consideration for this application is that the system must not significantly disturb the environment seen by the test article, as, for example, by presenting a warm spot in the otherwise uniformly cold wall surrounding the test article, or by presenting a source of glare that might confuse spacecraft optical components. This means that all heat sources, such as the drive motors, or cables that are dissipating heat, must be shrouded from the direct view of the test article. By the same token, shiny surfaces are not desired. The design must recognize these constraints.

A third design consideration is reliability and ease of maintenance. Since one of the major drivers for this program is the promise of decreasing the amount of down-time in testing, it would be counterproductive to have to halt testing to repair this equipment, or, when repairs or maintenance are required, to make it so awkward or time consuming to accomplish them as to defeat the purpose of installing the system in the first place.

A consideration notable for its absence from this list is extreme accuracy. Unlike most robotic applications, positional accuracy is not a strong requirement for the STAR system. To have position knowledge and repeatability accurate within a centimeter or two was judged to be quite adequate for this application. Since this is well within the capabilities of even the crudest mechanization, it was not a driver in the design.

PHASE 1 SYSTEM CONCEPT

A block diagram of the STAR Phase 1 system is given in Figure 2. It consists of the following assemblies:

a. The drive assembly,

b. The beam assembly,

c. The carriage assembly,

d. The pan/tilt assembly

e. The camera assembly

f. The in-chamber cable assembly,

g. The external cable assembly, and

h. The control console

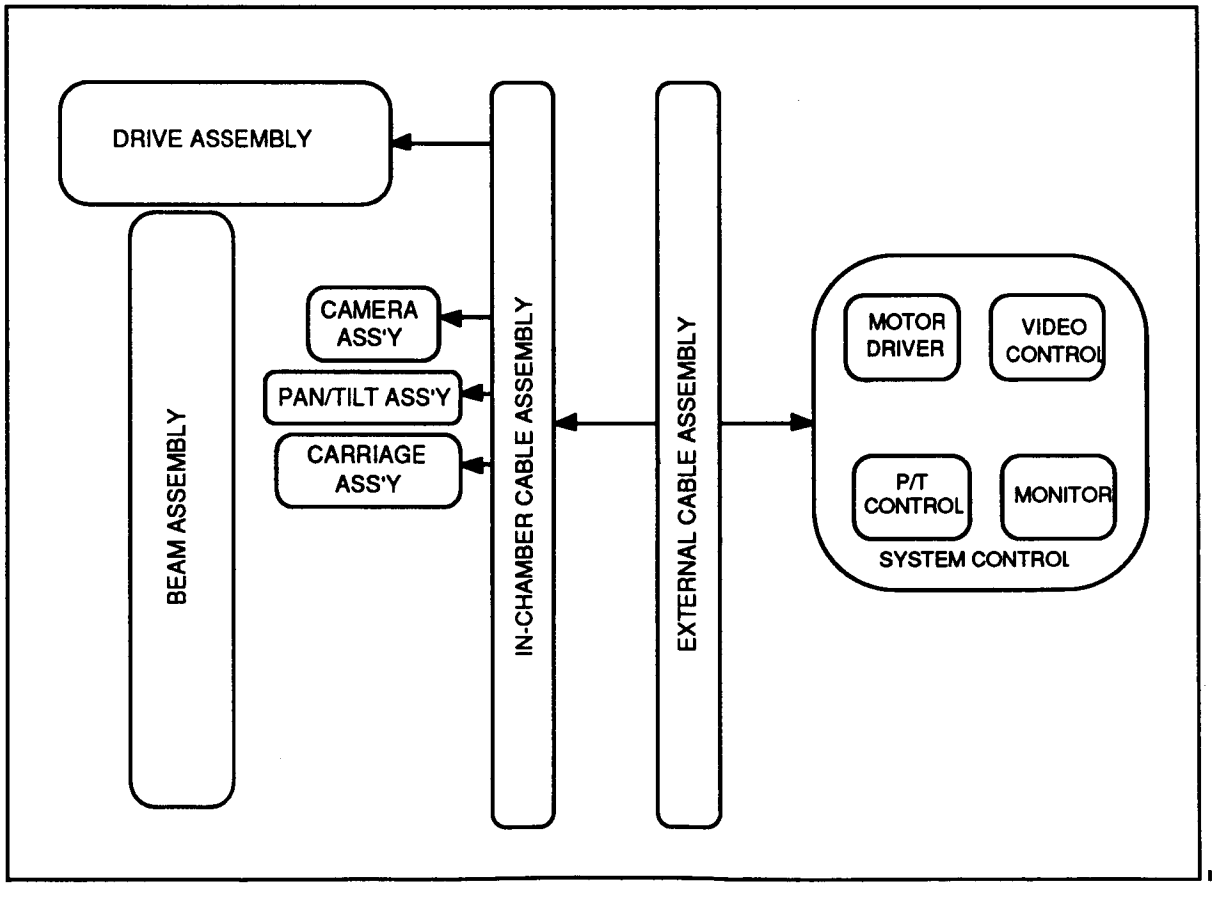


FIGURE 2. STAR SYSTEM BLOCK DIAGRAM

Table I

Comparison of Candidate Drive Mechanisms

MECHANIZATION / CHARACTERISTIC	RACK AND PINION	LEAD/BALL SCREW	CHAIN DRIVE	CABLE DRIVE	METAL BELT DRIVE
A. MATERIALS					
Properties at Cryo Temps	3	3	4	2	1
Properties over Wide Temp Range	4	4	4	2	1
Vacuum Rated	2	2	4	4	1
Non-Outgassing	2	2	3	4	1
Availability (Stock or Custom Comp)	2	4	3	3	3
Overall Mass	4	5	2	1	1
Comparitive Reliability	3	2	4	4	3
Est. Development Time	2	3	3	3	3
Simplicity of Assembly/Install.	3	4	3	2	1
Maintainability	3	4	4	3	2
Overall Cost	2	5	3	1	2
B. CLEANLINESS					
Lubrication Required	4	4	4	1	1
Rubbing vs. Rolling Contact	3	4	4	3	1
Ease of Cleaning	3	4	4	5	1
Debris Generation	3	3	4	3	1

1= BEST SELECTION 5= WORST SELECTIOI

Table II

Major Design Analyses Conducted

Rail/Linear Bearing		Beam/Roller
Open Belt		Closed Belt
Electrical Cable Spool		Rolling Loop
Pulley Diameter	vs	Belt Stress Belt Width Belt Thickness
Roller Diameter		Hertzian Contact Stress
Counter-Balanced		Non-Counter-Balanced

For each of these assemblies a number of design approaches were available; space does not allow a discussion of them all. In the following paragraphs the most significant design choices will be presented, with an indication of the rationale behind the choice. Emphasis is given to the in-chamber portion of the system, since that external to the chamber (external cabling and control console) presented no special design problems.

DESIGN DESCRIPTION

a. Drive Assembly

The drive mechanism is the heart of the design, and was the most difficult to design within the constraints imposed by the environment. Its design also drove the design of most of the other system assemblies. It will serve as an example of the kinds of analyses that were required to validate the detail design.

For the Phase 1 effort the final product is to be a vertically moving carriage on which will be mounted a pan/tilt platform. Any number of options were available to provide the vertical motion, including rack-and pinion, cable, chain, lead or ball screw, metal belt/pulley, etc. Each of these has its own peculiar advantages and disadvantages. Some of the most obvious are listed in Table I for various candidate mechanizations. Scanning Table I, and in light of the design considerations given above, the metal belt/pulley system appeared to provide the best match for the requirements. It is compatible with the design requirements of vacuum operability over an extreme temperature range and high cleanliness, it is potentially lightest in weight, and the simplicity of the concept makes its implementation appear reasonably economical. With little time to pursue in-depth trade-off studies, this approach was selected early as the baseline for STAR. Emphasis then shifted to the next layer of questions, i.e., what would it take to make it work. Questions of materials selection, accommodation for thermal expansion/contraction, unlubricated operation, avoidance of sliding contacts, cable accomodation, etc., required addressing in detail.

Table II lists some of the more significant design analyses required to come up up with a design that would meet all requirements. A detailed exposition of all the trade studies performed is beyond the scope of this paper; only the highlights and major conclusions will be reported here.

Drive Train

The selected motor is an Inland Brushless DC motor, Model RBE04500, flight and vacuum rated at 1000 in-oz of torque and for -55°C operation. It was selected specifically because it is designed for vacuum operation. A worm gear drive was selected because of its inherent non-backdriveability; while it violates the design criterion of no rubbing contacts, the rule was violated in this one case because it provided the additional advantages of a high gear ratio and smooth operation. The selection of materials (bronze for the worm gear and steel for the

drive gear) should minimize particulates generated by the rubbing motion. Even so, the drive will be totally enclosed in a housing to minimize the potential contamination that might be generated by this gearing.

Belt and Pulley

A primary concern was the selection of a belt material that could withstand the extremes of temperature and maintain high strength without embrittlement. An associated question was the diameter of the pulley, to minimize the bending stresses on the belt without violating the constraints of the maximum envelope allowed for the mechanism. Only solid belts were considered; woven belts, though far more amenable to the temperature extremes, were rejected because their (huge) surface areas posed too great a threat for contamination.

A survey of belt manufacturers uncovered no one who was willing to certify their belts for the specific environment we specified. However consultation with metallurgists indicated semi-hard 304 stainless steel should have the desired characteristics. A quick (though not highly scientific) test program involving flexing 304 stainless steel belts in liquid nitrogen and examining them for cracks verified this choice, which became the baseline. A further series of analyses on bending stress and yield strength as a function of bend radius and belt thickness led to the design parameter selection shown in Table III, which provides a design margin of about a factor of four for the maximum anticipated load of 100 lbs. However, to provide redundancy and an added margin of safety, a two belt system has been baselined.

TABLE III

Metal Belt Parameters

BELT LENGTH	320 INCHES
BELT WIDTH	2 INCHES
BELT THICKNESS	0.008 INCHES
BELT MATERIAL	304 SS COLD-WORKED 1/2 HARD
FATIGUE STRENGTH (100,000 CYCLES)	115,000 PSI @ 20°C 155,000 PSI @ -196°C
TENSILE STRENGTH	195,000 PSI @ 20°C 260,000 PSI @ -196°C
BELT CYCLIC LOAD CAP.	1400 LBS @ 20°C
BELT STATIC LOAD CAP.	3100 LBS @ 20°C

A final design decision was made to leave the belts open-ended. This automatically maintains belt tension as thermal expansion and contraction changes belt length, without the need for additional tensioning devices.

The baselined pulley/drive train system is shown in Figure 3, and, as an exploded view, in Figure 4. Figure 5 is a photograph of the actual hardware after initial assembly.

b. Beam Assembly

The selection of a belt and pulley drive allows the use of a simple and inexpensive U-beam for the vertical member. The beam is supported at the top only; the bottom is pinned via slotted holes to allow for thermal motion while maintaining verticality. The selected beam is 4 in. by 12 in., 6061-T6 aluminum, with 0.29 in. wall-thickness, black anodized, and for the initial demonstration is 25 feet long. Nothing in the design limits the beam to this length, however, and the basic design is fully adaptable to the larger 25' space simulator by using a longer beam.

The beam assembly is shown in Figure 6. The depth of the beam has been chosen to comfortably house the rolling loop cable that supplies power and signal connections to the carriage. A cover, split to allow passage of the wiring, is provided to shield the wiring from the chamber and to present to the chamber as uniform a temperature environment as possible.

c. Carriage Assembly

The primary design challenge for the carriage assembly was to provide free motion with minimum friction and no binding over the large temperature range and in high vacuum. As noted earlier, this eliminated from consideration any design that requires sliding contacts. The selected design, shown in Figure 7, provides six sets of wheels. Two sets (top rear and lower front) are load-bearing, while a second set (top front and lower rear) are spring loaded to assure contact is maintained. An additional set of wheels, also spring-loaded, is mounted to each side to maintain alignment as the beam changes dimensions during thermal cycling. Vespel has been selected as the wheel material to minimize the possibility of contamination by metal particulates that might be generated from the rolling contact of the wheels with the beam. Bearings are 440C stainless steel, and are unlubricated.

A major feature of the carriage is the ease with which the assembly can be removed. Loosening four bolts that secure the wheel assemblies, unfastening three electrical connectors, and pulling a single release pin from the belt yoke allows the entire carriage and instrument payload to be lifted off as a unit.

d. Pan/Tilt Platform

A pan/tilt platform previously used in the space simulator was available for at least temporary use with this system. This platform is shown in Figure 8.

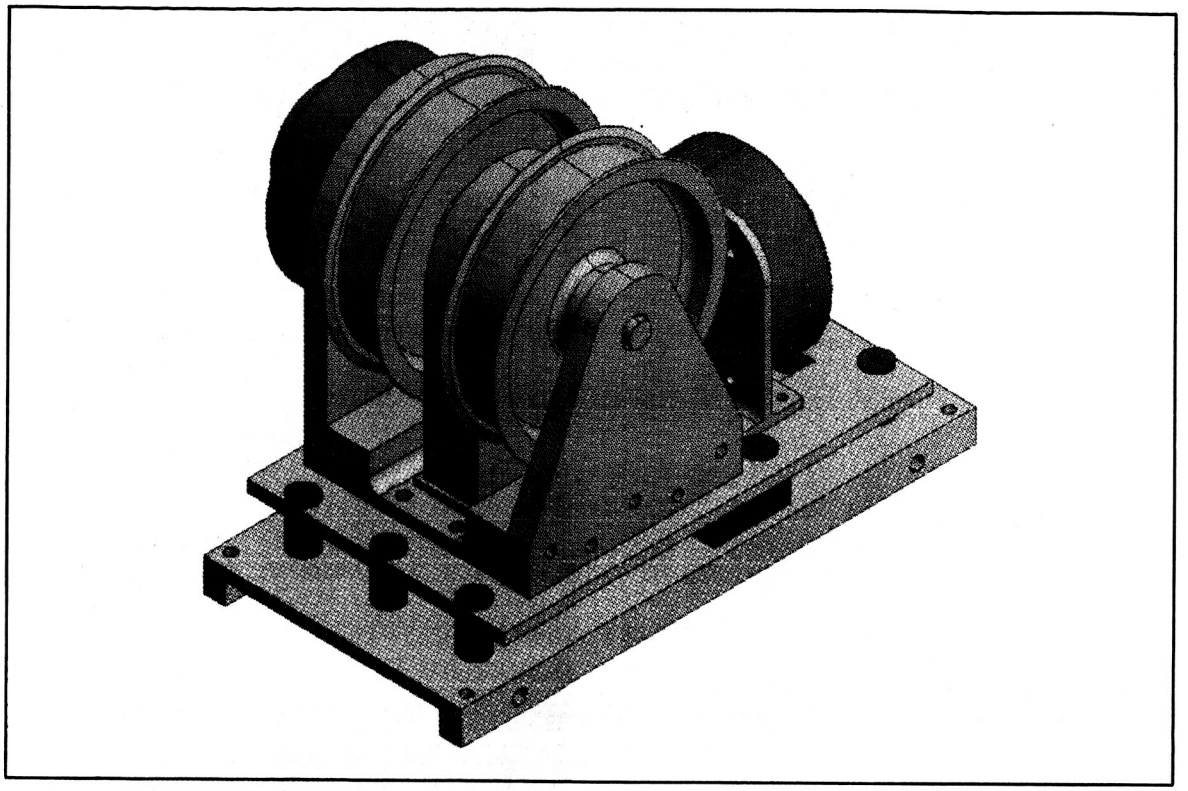


Figure 3. Perspective View of the Drive Assembly

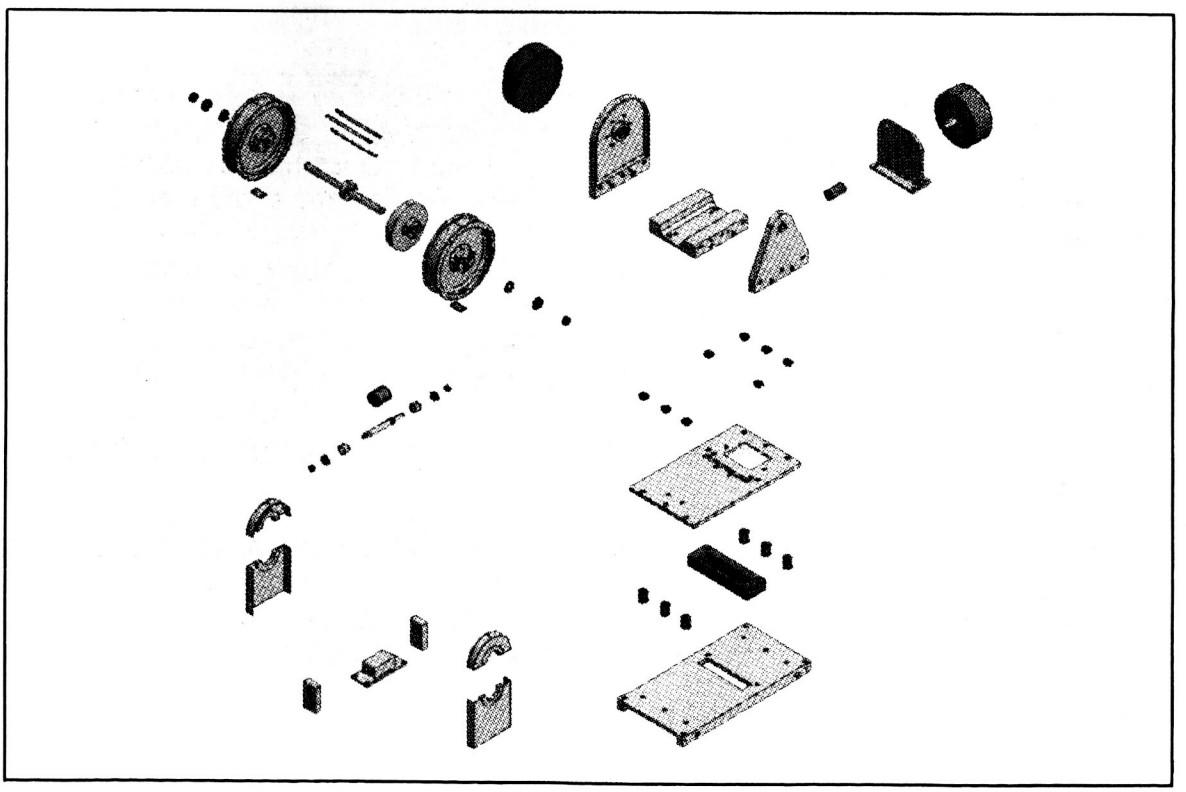


Figure 4. Exploded View of the Drive Assembly

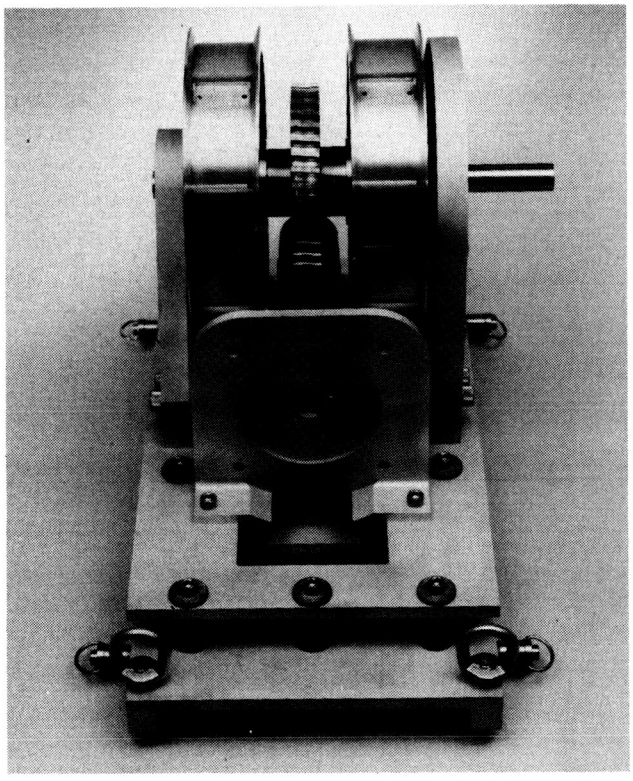


Figure 5. The Drive Assembly Hardware

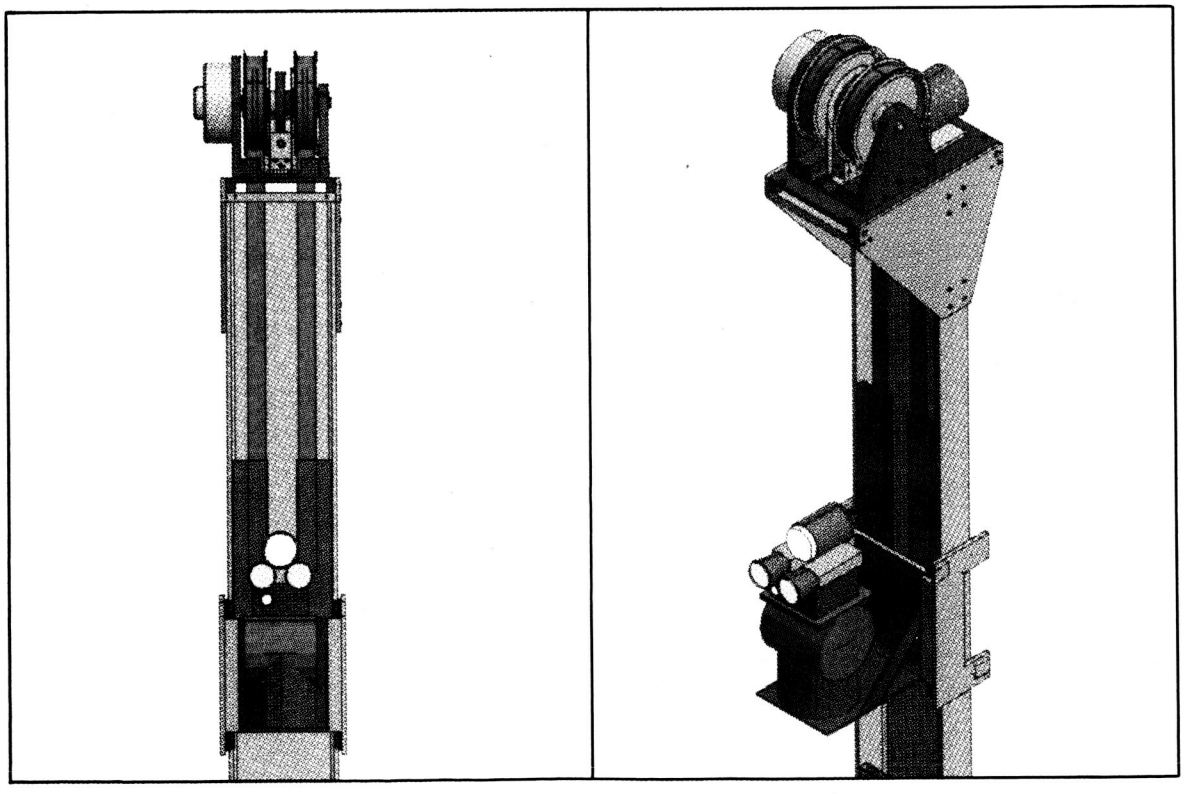


Figure 6. The Beam Assembly

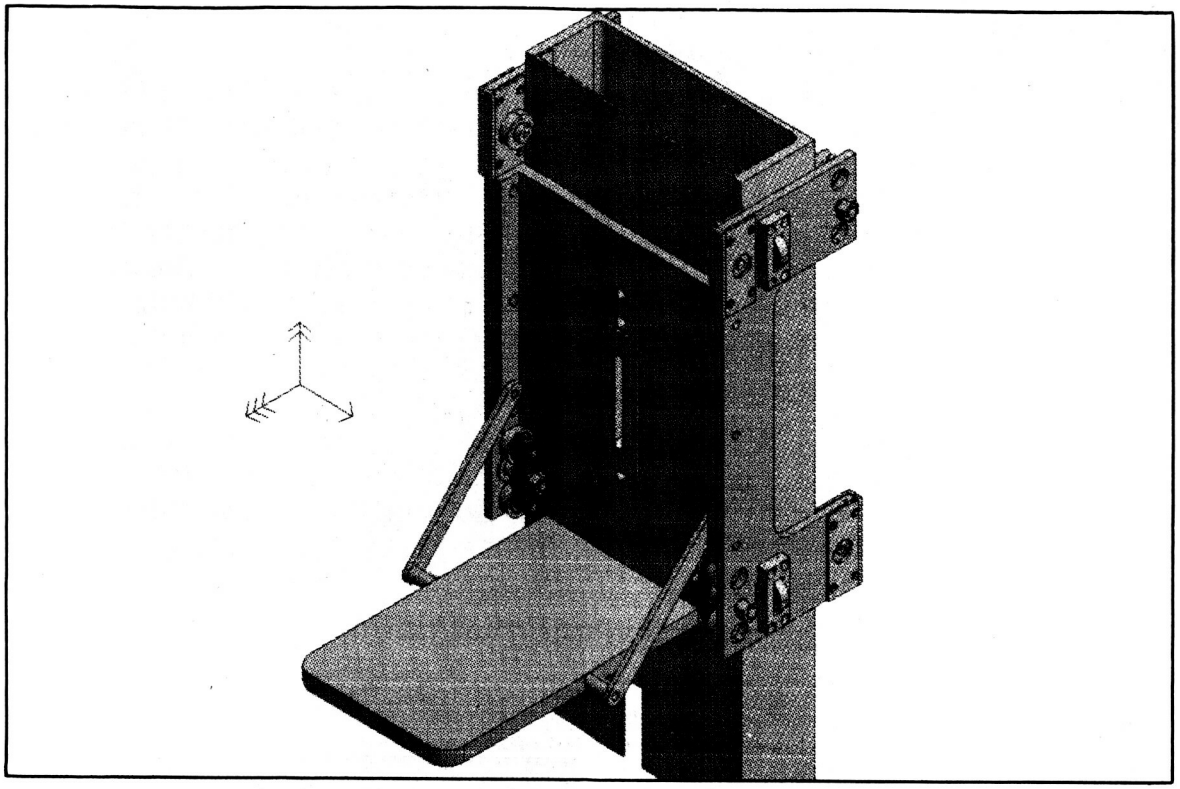


Figure 7. The Carriage Assembly

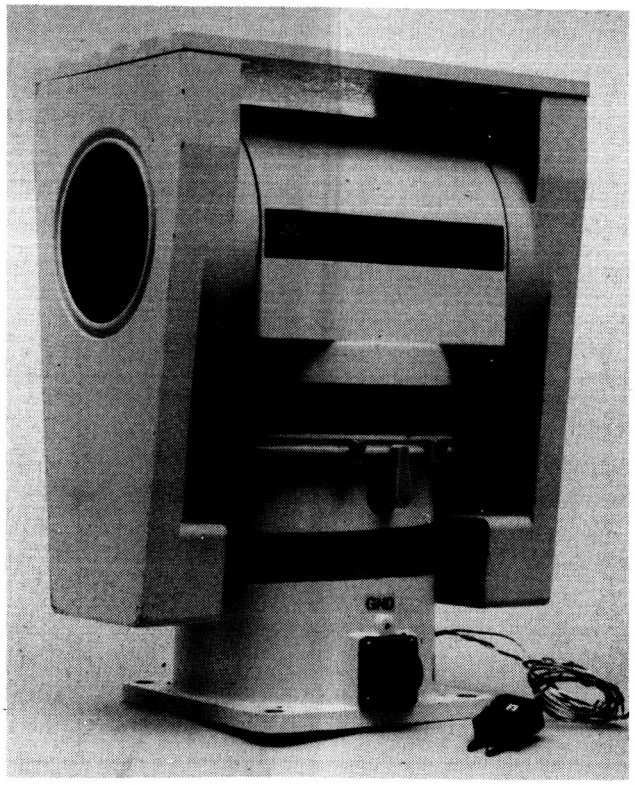


Figure 8. The Pan/Tilt Platform

e. Camera Assembly

The capability to be demonstrated in STAR Phase 1 is stereo viewing of the test article. To provide useful stereo over a reasonable range of target distances from the cameras, and to provide close-in viewing when required, a three camera arrangement, as shown schematically in Figure 9, taken from Ref 1, has been selected. The salient parameters for the camera arrangement for the Phase 1 demonstration are given in Table IV, also taken from Ref. 1. The cameras selected are designed to be vacuum operable, and will operate at temperatures as low as -50°C. To maintain them at this temperature will require thermal blanketing.

Cameras 1 and 2 have identical fixed focal length lenses, with manually adjustable focal distances. For any given test, these will be fixed prior to chamber evacuation. Camera 3's lens will be remotely adjustable for focal distance, and will serve as the best focused lens for both near and far stereo viewing.

The system will allow monocular camera viewing by any camera, and stereoscopic viewing by any pair of cameras. The video switcher will allow any camera image to be viewed on any of the three monitors.

A comprehensive discussion of the stereo system design process used here can be found in Ref. 2.

Table IV

Camera Configuration Parameters for the 10' Simulator

CAMERA PAIR	INTER-CAMERA DISTANCE	MONITOR #	MONITOR DIAG. SIZE	CONVERGENCE DISTANCE
1,2	5.6 INCHES	1	16"	2.2 METERS
2.3	2.5 INCHES	2	22"	1.6 METERS
1.3	3.1 INCHES	3	20"	3.0 METERS

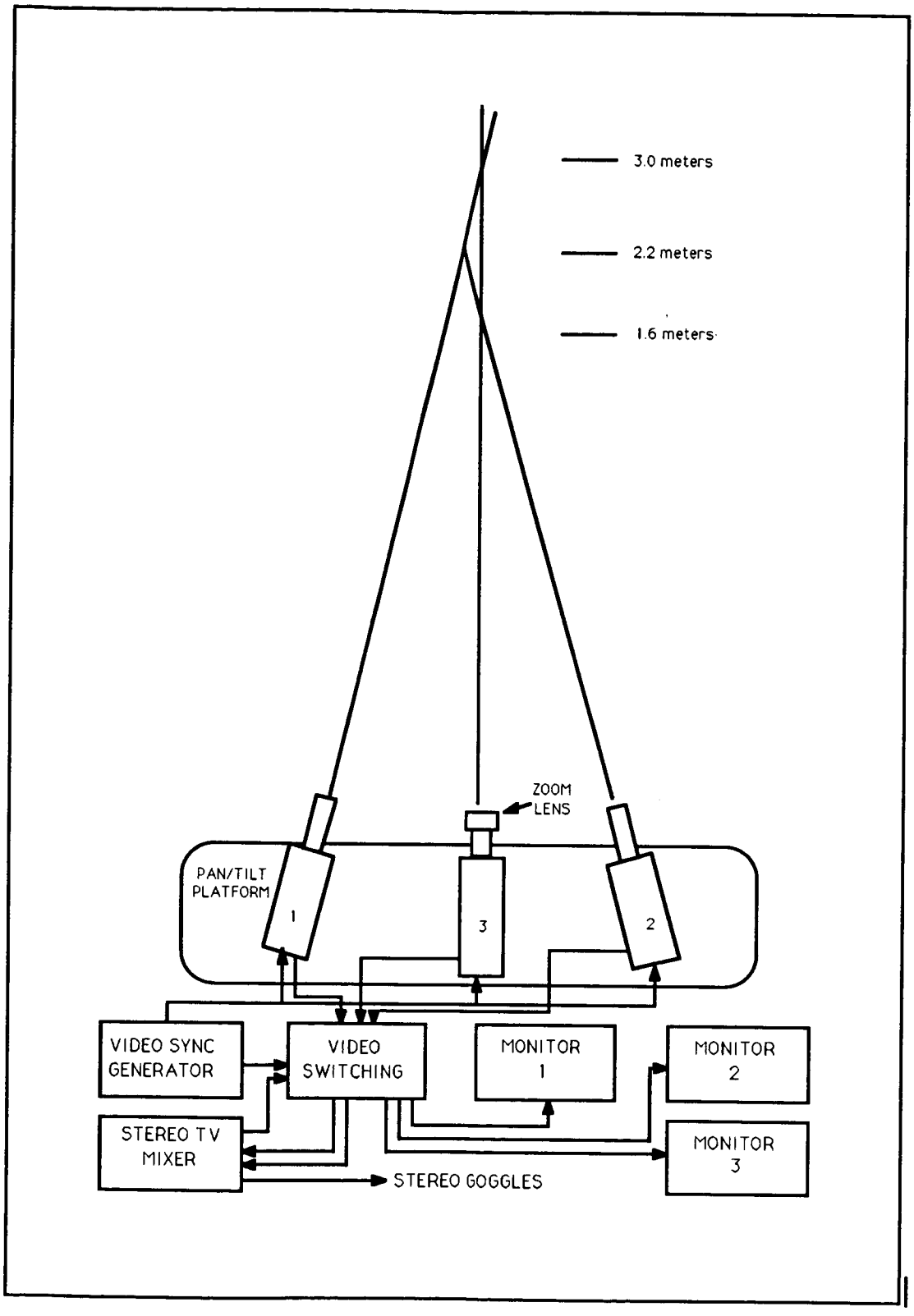


Figure 9. Camera Arrangement for the 10' Simulator

f. In-Chamber Cable Assembly

The overall schematic of the in-chamber cabling is shown in Figure 10. The only significant question here was the selection of cables for connecting the carriage and its payload to the chamber feed-throughs. To accommodate the motion of the carriage a rolling loop is employed. The cable, which is laid inside the beam, may reach temperatures close to -196°C. Again, no manufacturer was willing to guarantee the integrity of insulation flexing at such temperatures. Tests were run, flexing various flat cables with insulation rated for vacuum operation in a liquid nitrogen bath. In this testing, teflon insulated cables proved to have quite satisfactory flexing properties at LN_2 temperatures, and are being used for all in-chamber cables.

g. External Wiring Assembly

The external wiring poses no particular problem, and is mentioned here only for completeness. The external cabling is shown schematically in Figure 11.

h. Control

A 386 microprocessor is being built into the STAR system even though the requirements of the Phase 1 demonstration could be satisfied with a far less capable controller. The controls for the Phase 1 demonstration are relatively simple: vertical position commands are given via keyboard input and the motor encoder provides the necessary position feedback. Software limits are incorporated, and are backed up by mechanical limit switches. The pan/tilt unit is run on the same principle. The focal length of the lens system on camera 3 is controlled open-loop by the operator.

Later phases of the program will see more complex control loops incorporated.

STATUS

Table V summarizes the status of each of the assemblies of the STAR system; the present schedule calls for the system to be installed in the 10-ft SS by the end of August, and for the first full-up demonstration of the system by the end of September.

REFERENCES

1. Diner, D. B. "Stereo Viewing System for STAR" JPL IOM 3474-92-059, June 30, 1992

2. Diner, D. B., and D. H. Fender "Human Engineering in Stereoscopic Viewing Devices" JPL Report #D-8186, January 15, 1991

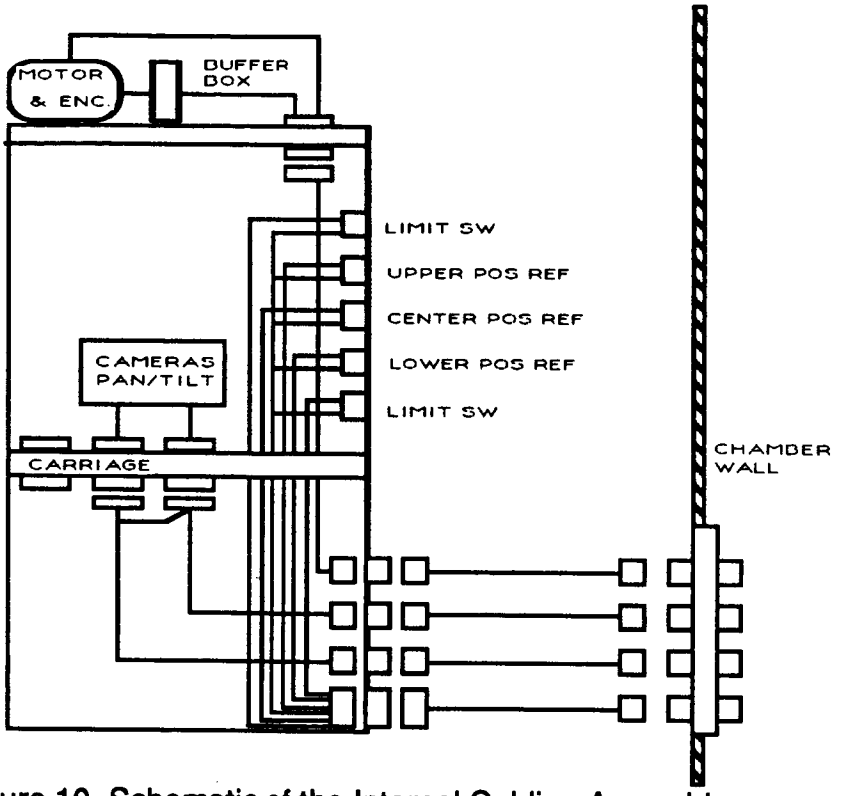


Figure 10. Schematic of the Internal Cabling Assembly

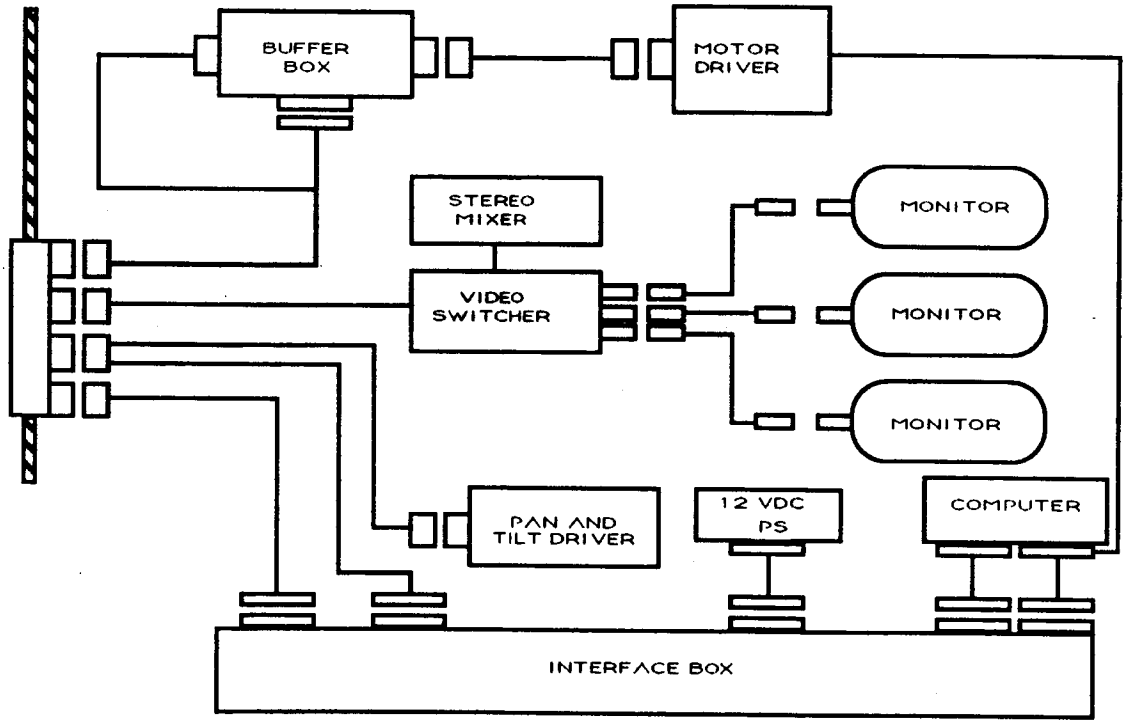


Figure 11. Schematic for the External Cable Configuration

Table V

Status Summary

	Detail Design/ Proc. Spec.	Fabricate Procure	Deliver	Assemble/ Checkout
Drive	Complete	Complete	Complete	Complete
Drive Motor	Complete	Placed	Due Aug 14	
Carriage	Complete	In Process	Due July 31	
Beam	Complete	Complete	Complete	Complete
Pan/Tilt	Complete	Existing		Complete
Cameras	Complete	Placed	Due Aug.3	
Internal Cabling	Complete	In Process	Due Aug 15	
External Cabling	Complete	In Process	Due Sept. 1	
Control Console	In Process		Due Sept. 1	

TeleOperator/telePresence System (TOPS) Concept Verification Model (CVM) Development

Mike S. Shimamoto
NAVAL COMMAND, CONTROL AND OCEAN SURVEILLANCE CENTER
RDT&E Division (NRaD) Hawaii Detachment
PO Box 997
Kailua, Hawaii, 96734 U.S.A.

ABSTRACT

The development of an anthropomorphic, undersea manipulator system, the TeleOperator/telePresence System (TOPS) Concept Verification Model (CVM) is described. The TOPS system design philosophy resulting from NRaD's experience in undersea vehicles and manipulator systems development and operations is presented. The TOPS design approach, task teams, manipulator and vision system development and results, conclusions, and recommendations are presented.

INTRODUCTION

A major step has been taken toward the development of an advanced, telerobotic, undersea work system with the TeleOperator/telePresence System (TOPS) Concept Verification Model (CVM) (Figure 1). The long term objective of the TOPS program is to develop the technologies required to build remote work systems that are functionally equivalent to a diver in performing unstructured undersea tasks. Such a remotely controlled manipulator system would not be constrained by the diver's operational limitations in hazardous areas, great ocean depths, cold temperatures, and submerged operating time. The emphasis of the project is on developing the capability for performing tasks that require the dextrous, adaptive, and judgmental capabilities of man rather than on performing precise, well-defined tasks that can be addressed by purely robotic systems or specialized tools.

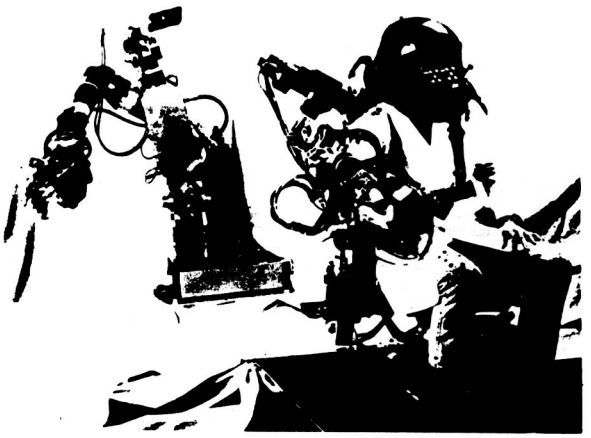


Figure 1. TOPS CVM

BACKGROUND

Organizations contributing to the development of the TOPS CVM and their areas of expertise are as follows: NRaD (US Navy remotely operated vehicle and manipulator development); Sarcos, Inc. (SI) and the Center for Engineering Design (CED) at the University of Utah (dextrous hand/arm, entertainment robots, and robotic component development); and Armstrong Aerospace Medical Research Laboratory (AAMRL) at Wright-Patterson Air Force Base and Technology Innovations Group (TIG) (helmet mounted display vision systems development).

Teleoperator Systems Development at NRaD

Over a span of two and a half decades, the Advanced Systems Division of NRaD's Hawaii

Detachment has developed manned undersea vehicles; unmanned, remotely operated undersea vehicles (ROVs); unmanned remotely operated ground vehicles (UGVs); and teleoperated manipulator systems[1,2,3].

During the development of the Remote Unmanned Work System (RUWS) (Figure 2) and several other ROVs, test operations were conducted in recovery, inspection, and emplacement tasks (Figure 3). The "lessons learned" from those operations provided impetus to the TOPS program.

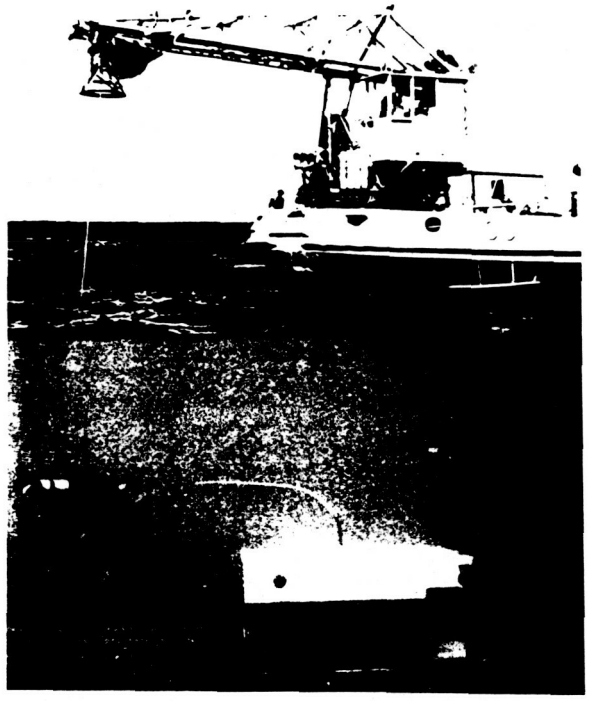


Figure 2. RUWS.

Although a set of hydraulic tools had been designed and fabricated for use by the RUWS manipulator, additional special tools were often required for new tasks (Figure 4). During several operations, the tools had to be modified or new tools fabricated, because the task was not quite as it was "supposed to be." Although, lots of pre-operations planning were done and special configurations for the vehicle were implemented, few missions were completed without difficulty. Navy salvage operations, by nature, usually have many "unstructured" tasks when recovering wreckage and items from wreckage.

Figure 3. Manipulator work.

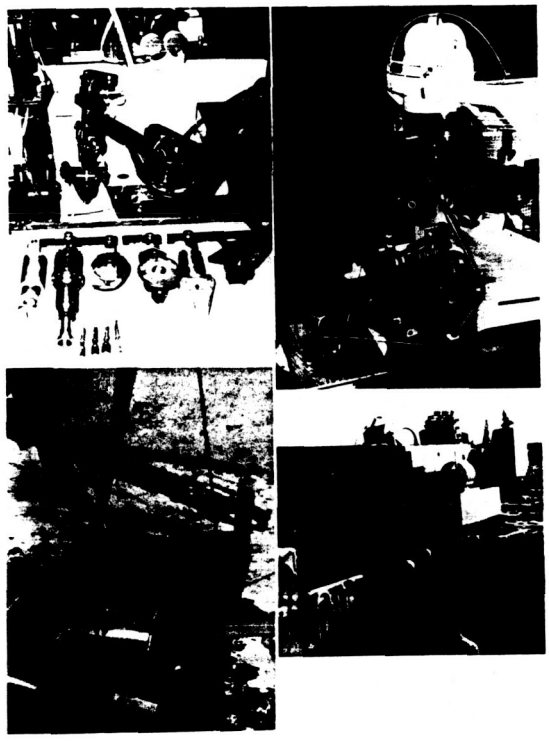


Figure 4. Vehicle configuration.

Simple diver tasks, such as putting a snap hook onto a shackle, proved to be difficult because of the limitations in dexterity of the manipulator and mobility of the vehicle. If currents were present, the object to be worked on was approached with the vehicle heading into the current; this frequently resulted in an orientation to the task that was less than ideal for the manipulator. Maneuvering the vehicle for proper positioning usually resulted in agitation of the bottom sediment, which obscured the remote operator's visibility. Conditions such as

those for each mission seemed to provide unique challenges to the operators even when the missions consisted of fairly simple tasks. The operators were often more frustrated than fatigued in attempting to complete the tasks for a successful mission. Tasks that could easily be performed by divers were not at all trivial for an ROV work system. These lessons indicated that a diver-equivalent work system might provide the work capability needed for many undersea missions where present ROV and manipulator systems are inadequate. The capabilities of such a system could also be applied to other hazardous missions on land and in space.

Diver Tasks

An assessment of tasks performed by Navy and civilian divers determined: (1) the importance of various tasks within dive missions, (2) the manipulative and sensing capabilities used by the divers to perform the missions, and (3) the key design parameters for the development of a diver-equivalent manipulator system.

In determining the importance of various tasks within dive missions, it became clear that the major differences between what divers could do and what could be done with manipulators were that divers could perform a series of complex tasks and adapt to the differing tasks to successfully complete a mission. The divers used their own manipulative and sensing capabilities that were required to complete the tasks. Maneuverability, dexterity, and full sensory capability were key to the adaptability and versatility required to successfully complete the variety of tasks within the missions.

In determining the key design parameters for a diver-equivalent manipulator system, it became evident that the best configuration that would allow an operator to perform like a diver was a system configured the same way as the human operator (i.e., an anthropomorphic configuration). A manipulator system with joints and links that matches the operator's (kinematic equivalent) and with all manipulative appendages and sensory systems in the same relative positioning (spatial correspondence) as the operator's appendages and sensory systems would allow the operator to perform the tasks as if he/she were present at the work site.

A system that maintains spatial correspondence between the slave and the operator allows the operator to use his/her experiences from infancy to the present. If spatial correspondence is lost, people can adjust, but only by sacrificing performance. The loss in performance shows up in objective measures such as additional training required to attain proficiency, higher error rates, longer times to complete the tasks, as well as increased mental and muscular fatigue by the operators[4].

Anthropomorphic Manipulator Development

The first anthropomorphic (human configured) manipulator developed at NRaD was the Remote Presence Demonstration System[1,2] (nicknamed "Greenman"), shown in Figure 5. It was assembled in 1983 using MB Associates arms and an NRaD-developed torso and head. It had an exoskeletal master controller with kinematic equivalency and spatial correspondence in the torso, arms, and head. Its vision system consisted of two 525-line video cameras each having a 35° field of view and video camera eyepiece monitors mounted in an aviator's helmet.

Figure 5. Remote Presence Demonstration System.

Greenman provided NRaD with valuable experience in teleoperation and telepresence issues and designs. Even with its simple claw hands and no force or tactile feedback, novice operators could readily perform manipulative tasks without training. However, it clearly showed that dextrous hands, force feedback, and a high-resolution vision system were necessary for diver-equivalent work capability. Also, the Greenman was not designed for in-water use, and

demonstrations of in-water work was deemed necessary to fully demonstrate the diver-equivalent concept.

TOPS PROGRAM DEVELOPMENT

TOPS Long-Term Concept

The long-term concept for a diver-equivalent manipulator system is shown in Figure 6. The master controller "fits" the operator like a business suit and senses his/her hand, body, and head motions. The slave manipulator mimics the operator's motions, senses its interaction with the environment, and provides sensory feedback to the operator via the master controller in a manner natural to him/her.

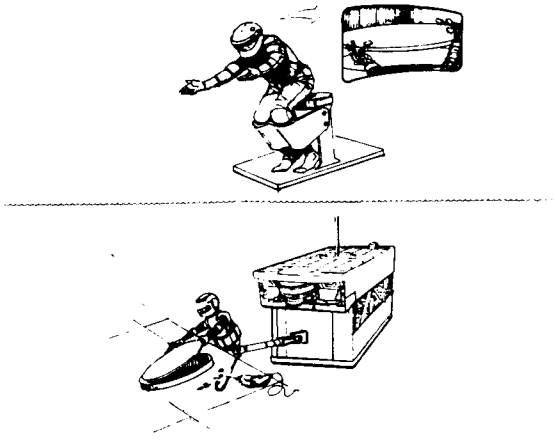


Figure 6. TOPS Concept.

An assessment was conducted of available, near-term, and long-term technologies in planning for the development of the first TOPS model to verify the concept. Because the first model would be a 3-year project only, long-term technologies were not included in the project scope.

Long-term technologies identified for future TOPS systems were: (1) tactile telepresence systems, (2) high-definition TV (HDTV), (3) human equivalent dextrous hands, (4) the integration of virtual reality with the vision system, (5) advanced manipulator controllers, and (6) passive sonar for underwater directional hearing .

TOPS CVM

The first model of TOPS was called the Concept Verification Model (CVM). This model incorporated available and near-term teleoperation and telepresence technologies including (1) dextrous hands, (2) high-fidelity force feedback, (3) high-resolution head-coupled vision, and (4) an integrated, natural master controller with spatial correspondence. The major thrust of the technologies was in the development of the two major subsystems: (1) the manipulator and (2) the vision system.

TOPS CVM Manipulator Development

The development of the TOPS CVM manipulator was contracted to Sarcos, Inc. and the Center for Engineering Design at the University of Utah. The hand was developed in the first phase; the arm was developed and then integrated to a revised hand in the second phase; and the torso and head were developed and integrated in the third phase. The supporting control system was developed throughout all phases.

In the first phase, the hand development consisted of finger, hand, and wrist design concepts; tendon, actuator, and valve evaluation and development; sensor and supporting structure development; and antagonistic (pull-pull) servo control system development. A brassboard, 9-degree-of-freedom (DOF) hand was developed incorporating a 4-DOF thumb, a 3-DOF index finger, and a 2-DOF middle finger (Figure 7). The hand was attached to a 3-DOF wrist incorporating coincidental axes. The exoskeletal hand master represented a major design breakthrough where the structure fit on the backside of the hand but had virtual joints that matched the operator's finger joints. The brassboard hand was demonstrated at the end of the first phase (1 year). Demonstrations showed that the hand had the capability to perform standard hand grasps and manipulate various objects (such as threading a #10 nut onto a stud, and grasping and using standard hand tools), and showed high-sensitivity force feedback with high inter-system stiffness.

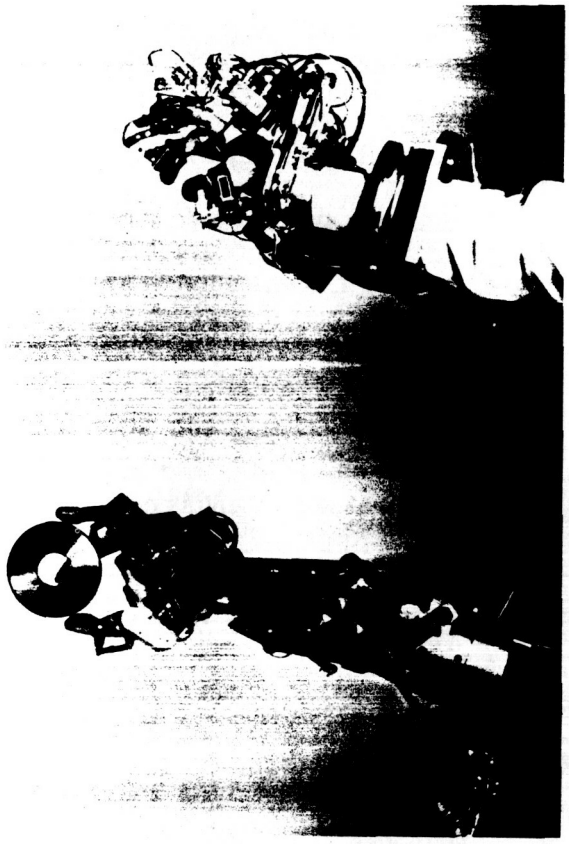


Figure 7. TOPS CVM brassboard hand.

In the second phase, the hand was revised while the arm was developed, then the arm and hand were integrated; low-friction rotary actuators were developed; and development of high-performance servo system components and controllers was continued. The arm was designed with a 3-DOF shoulder and 1-DOF elbow. The 3-DOF shoulder was designed to allow forearm/elbow orientations for various work task requirements. The exoskeletal arm master allowed full, natural operator control of the slave manipulator.

In the third phase, the torso and head were developed; subsystem and component development of valves, actuators, tendons, sensors, and hand designs were continued; all subsystems were integrated; then the system was tested in water. The 3-DOF torso was developed to provide a natural, short-range mobility and repositioning platform for the arm and vision. The 3-DOF head was developed to provide natural, spatially correspondent visual positioning capability. Force feedback was not incorporated in the torso and head.

TOPS CVM Vision

The development of the vision system capitalized on the efforts by AAMRL on helmet mounted display (HMD) systems for the US Army's Light Helicopter, Experimental (LHX) program. After evaluating HMD prototypes for the LHX, a "pancake window" HMD configuration was selected for TOPS and a contract was awarded for an HMD to Technology Innovations Group (TIG) of New York. The HMD included a pair of 1023-line, monochrome CRTs with 68° field of view optics (approximately the view from a diver's mask); air cooling for comfort; and a "clamshell" rear-hinged section to make it easy to put the helmet on and take it off (Figure 8).

The remote portion of the vision system consisted of a pair of 1023-line monochrome cameras with fixed-focus lens mounted in an underwater housing.

A sophisticated display electronics package was acquired from AAMRL. The display electronics (developed for the LHX program) allows precise distortion correction for each channel, video signal, and CRT display. The correction parameters for each item can be stored on disk to allow rapid component changeout and reconfiguration.

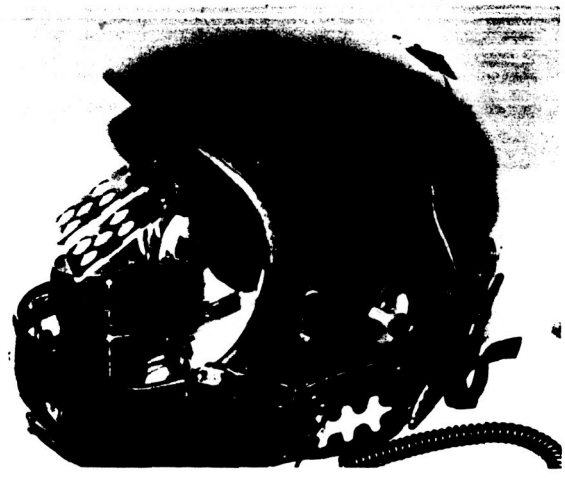


Figure 8. TOPS CVM Helmet Mounted Display.

TOPS CVM Overall Objectives Met

The overall TOPS CVM technical objectives were met in the development of an advanced manipulator system that begins to approach diver work capability. A high dexterity (22 DOF) manipulator with high-fidelity force feedback and a high-resolution, head coupled, stereo vision system was achieved. The combination of high dexterity that is kinematically equivalent to the operator, good force reflection, and a spatially correspondent 3-D vision system contributes to a high level of telepresence, i.e., the perception that the system is transparent to the operator. The operator feels that he/she is at the work site performing the task, and can concentrate on the task and not on operating the system.

Lessons Learned

Very valuable lessons were learned during the development and testing of the TOPS CVM[5]. The manipulator demonstrated great potential for performing a variety of manipulative tasks. The force reflecting exoskeletal system was natural and easy to use. However, subtle differences in kinematics and materials had major impacts on system performance. When link lengths and joint axes of the master controller did not properly match the operator's links and joints, and when grasping and positioning were not replicated exactly, the operator usually worked with significantly more caution and at a reduced speed. The fingertip configuration and materials of the slave hand also impacted the ability to securely grasp objects and, hence, the operator's confidence and speed of task performance. The compensation for gravity in the hand and arm for all areas of the workspace is very important to overall system performance and in the minimization of operator fatigue. Also, the capability to freeze operator selected joints would be very valuable for fine positioning tasks.

The tendon system proved too delicate, bulky, and complex for underwater operational systems. Tendon technologies that more closely replicate the human tendon system need to be developed.

The torso proved very useful in extending the manipulator's work volume and capability, in changing the viewing perspective, and in providing a "natural zoom" capability (the ability to position the cameras closer to the work task simply by leaning toward the object).

CONCLUSIONS/RECOMMENDATIONS

Telerobotic systems will continue to be important for environments and tasks that are hostile to humans, but where man's cognitive and manipulative capabilities are needed. This case is particularly true for accidents where explosives, chemicals, nuclear materials, extreme heat or cold, etc., would expose humans to great danger. Accidents also present the high probability of occurrence of unstructured tasks that need to be performed to accomplish the mission.

Unstructured tasks usually require that full manipulative, sensory, and cognitive capabilities be employed. Any manipulative or sensory capability that a manipulator system does not provide is a "handicap" to the operator. The TOPS CVM represents a giant step taken towards minimizing the "handicaps" an operator inherits with a typical manipulator system.

However, as discussed in the section on Lessons Learned, continued refinements are needed in the TOPS CVM design to improved operator machine interfaces and produce a ruggedized, smaller hand for an operational system.

The next development phase requires continued developments in component technologies for increasing hand dexterity, providing underwater directional hearing capability, enhancing vision, and providing tactile telepresence.

Component development required for increased hand dexterity include reliable, low-stiction tendons, biological-like lubricants, and compact tendon routing technologies; small, responsive, lightweight, muscle-like actuators; finger- and palm-padding type material; and tough skin-type material.

The development of small, high-definition TV cameras and monitors are needed for 20/20 color vision systems.

REFERENCES

1. HIGHTOWER, J.D. and SMITH, D.C., "Teleoperator Technology Development," Proceedings of the 12th Meeting of the United States-Japan Cooperative Program in Natural Resources, San Francisco, CA, 1983.

2. HIGHTOWER, J.D., SMITH, D.C., and WIKER, S.F., "Development of Remote Presence Technology for Teleoperator Systems," 14th Meeting of the United States-Japan Natural Resources Committee, September 1986.

3. MURPHY, D.W., "Advances and Experience with Teleoperated Systems Incorporating Remote Presence," 17th Meeting of the United States-Japan Cooperative Program in Natural Resources, May 1991.

4. PEPPER, R.L., "Human Factors in Remote Vehicle Control," 1986 Annual Meeting of the Human Factors Society, 1986.

5. SMITH, D.C. and SHIMAMOTO, M.S., "TeleOperator/telePresence system (TOPS) Concept Verification Model (CVM) Testing and Demonstration, Final Report," unpublished, July 1991.

FIRE PROTECTION FOR LAUNCH FACILITIES USING MACHINE VISION FIRE DETECTION

Douglas B. Schwartz

Air Force Civil Engineering Support Agency
Tyndall Air Force Base, Florida

INTRODUCTION

Fire protection of critical space assets, including launch and fueling facilities and manned flight hardware, demands automatic sensors for continuous monitoring, and in certain high-threat areas, fast-reacting automatic suppression systems. Perhaps the most essential characteristic for these fire detection and suppression systems is high reliability; in other words, fire detectors should alarm only on actual fires and not be falsely activated by extraneous sources. Existing types of fire detectors have been greatly improved in the past decade; however, fundamental limitations of their method of operation leaves open a significant possibility of false alarms and restricts their usefulness.

At the Civil Engineering Laboratory at Tyndall Air Force Base in Florida, a new type of fire detector is under development which "sees" a fire visually, like a human being, and makes a reliable decision based on known visual characteristics of flames. Hardware prototypes of the Machine Vision (MV) Fire Detection System have undergone live fire tests and demonstrated extremely high accuracy in discriminating actual fires from false alarm sources. In fact, this technology promises to virtually eliminate false activations. This detector could be used to monitor fueling facilities, launch towers, clean rooms, and other high-value and high-risk areas. Applications can extend to space station and in-flight shuttle operations as well; fiber optics and remote camera heads enable the system to see around obstructed areas and crew compartments. The capability of the technology to distinguish fires means that fire detection can be provided even during maintenance operations, such as welding.

CURRENT FIRE DETECTION TECHNOLOGY

Fire detectors used today sense smoke, heat, or electromagnetic energy such as ultraviolet (UV) or infrared (IR) emissions. Only the latter type, also known as optical fire detectors (OFDs) are capable of speed-of-light sensing of flames; thus, they are employed where fast, remote sensing is required. Flames emit characteristic

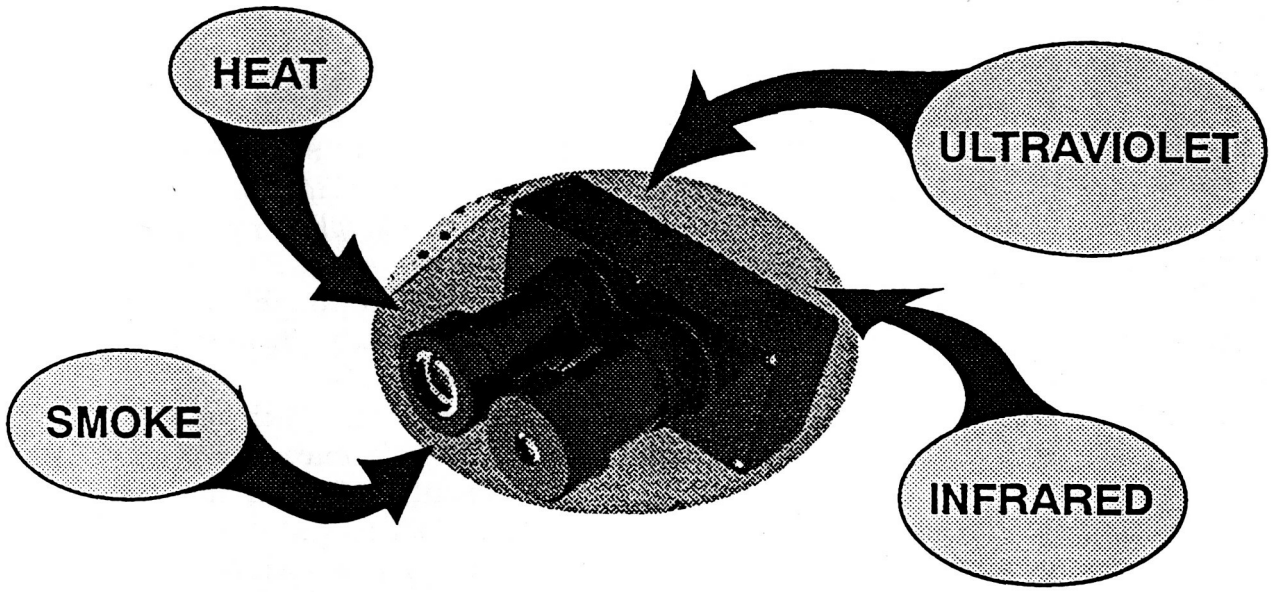


Figure 1. Conventional Means of Fire Detection (UV/IR detector illustrated).

electromagnetic emissions in particular bands, specifically the 0.18-0.24 micron band for UV and the 4.4 micron CO2 band for IR. However, any source emitting these frequencies will cause the detector to alarm. Ultraviolet detectors, for example, are commonly set off by reflected sunlight and arc welding. Infrared detectors can be set off by hot exhaust manifolds on powered support equipment, propane torches, and other heat sources. Many installations still use these single-band detectors, such as launch towers. In many cases, the inherent unreliability of this detection method has led to to disconnection from automatic suppression systems. To improve reliability, mulitispectral detectors have been introduced in the past few years, which require the presence of both UV and IR sources or two discrete, characteristic infrared frequencies. Although false alarms with this type have been greatly reduced, multiple sources of UV and IR radiation, often found in complex environments, can still cause false alarms. Detectors have also malfunctioned due to the presence of X-rays from testing equipment, vibration, and other hazards of the operational environment. The Civil Engineering Lab is completing testing of optical fire detectors against false alarm sources. The result will be a military standard to allow manufacturers to produce more false alarm resistant and environmentally hardened systems.

No matter how well optical fire detectors are constructed, the nature of ultraviolet/infrared detection implies certain fundamental limitations. Optical fire detectors trade off speed for accuracy; the faster the system is set to detect a fire, the higher the false alarm rate. OFD's are capable of detecting fires in less than 1/100 second, but are typically slowed to 3-30 seconds detection speed. This can be a significant delay where fast response time is needed, such as protection of heat-sensitive composite aircraft like the B-2 bomber. Since OFD's only sense the magnitude of absorbed energy impacting the detector, they cannot judge the absolute size of a fire; a small fire close up emits the same energy to the detector as a large fire further away. Intensity of UV or IR energy reaching the detector drops off rapidly with the inverse square law, leading to a maximum reliable range of about 120 feet. This is a serious limitation for coverage of large spaces, such as warehouses.

To circumvent these limitations, an effort was initiated in 1990 to incorporate image processing technology into a new type of fire detector. Machine vision fire detection actually "sees" the fire in the visible spectrum and applies numerous and flexible criteria to judge the presence of fire. The detection process has been designed to assure immunity to all known sources of false activations while reliably and rapidly detecting visible flames. Furthermore, the nature of the system means it can be adapted to visually sense non-fire threats, such as fuel vapor clouds.

The front end of the system is a solid-state video camera, which uses a CCD (charged-coupled device) to convert light into electronic information. The CCD consists of a square grid of picture elements (pixels) typically 512 on a side, or over 250,000 pixels total. This chip can resolve a one square foot fire at 100 feet. The intensity of red, green, and blue light impacting each pixel is sequentially input into computer memory, which builds up a "virtual image" or frame which can then be analyzed. This takes place every 1/30 second.

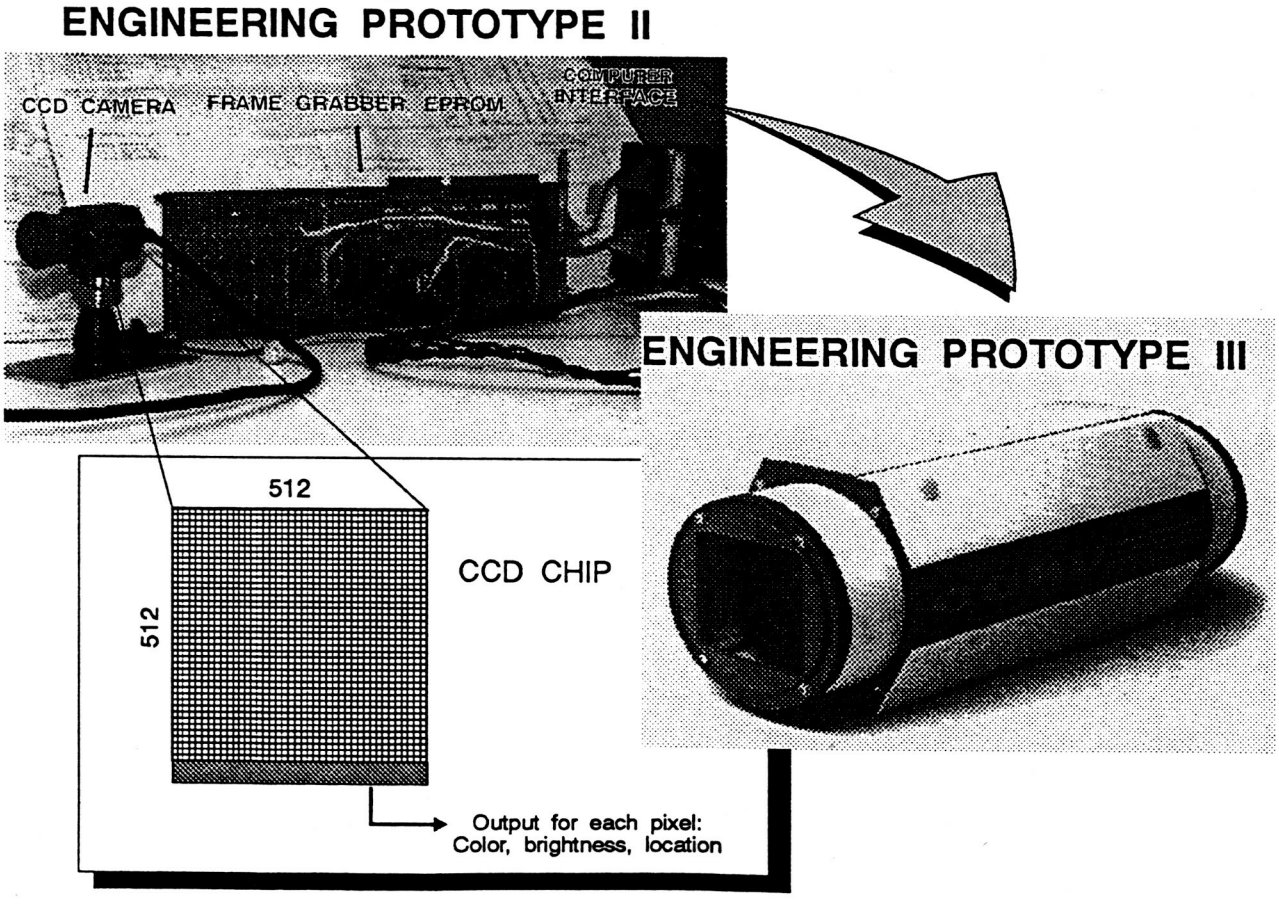


Figure 2. Machine Vision Hardware.

Pixels are checked for minimum brightness (intensity) and color within red, green, and blue parameters. Succeeding frames are compared, revealing changes in color from frame to frame, behavior of the edge of the object, and growth rate. Actual

fires exhibit rapid color changes from frame to frame, have highly variable edges, and tend to grow outward from a starting point.

Size of the fire is computed by counting the pixels meeting the "fire" criteria. Where installed in a fixed setting, such as a launch tower, the system will be calibrated at installation to relate position within its field of view to a particular size. Thereafter, the pixels across the base of the fire can be summed and actual size computed from the number of pixels from the lower edge of the field of view. Portable or mobile installations will use two cameras on a known baseline for range estimation.

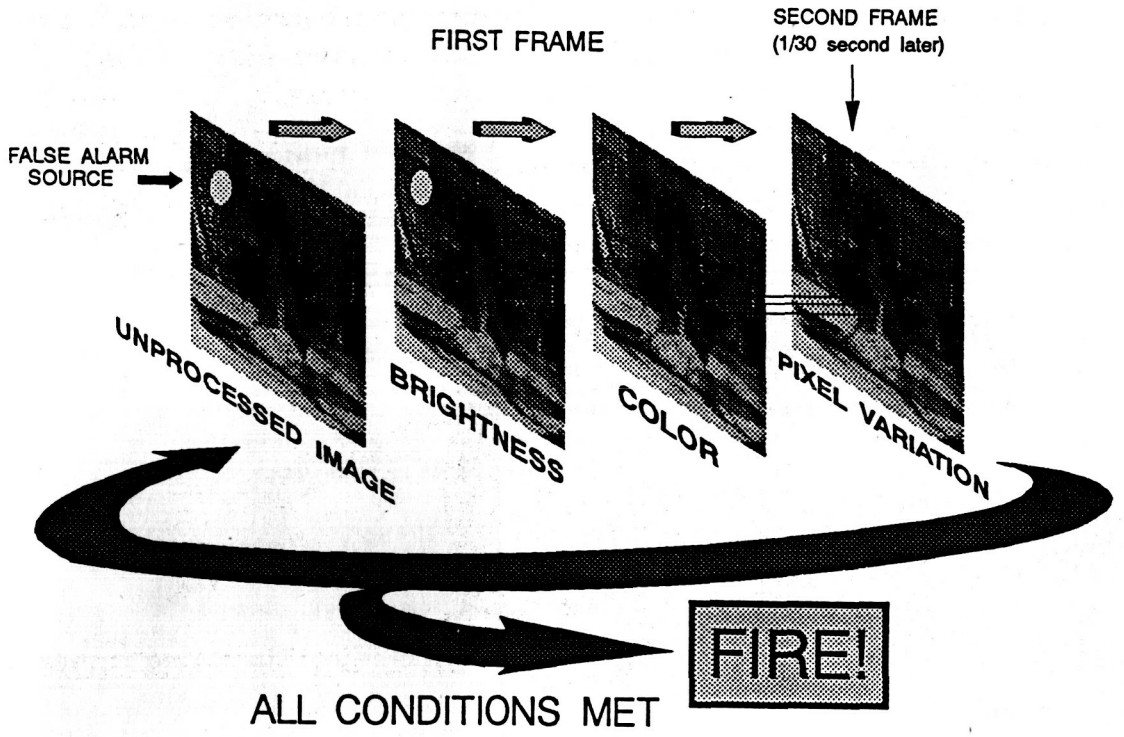


Figure 3. Fire Decision Criteria.

Knowledge of the actual size and growth rate allows determination of the degree of threat of a particular fire. A detection and suppression system can have a selection of possible responses depending on the threat, instead of the current all-or-nothing approach. In a typical Air Force application, for example, machine vision detectors would be linked to an automatic suppression system capable of dispensing tens of thousands of pounds of firefighting foam onto a hangar floor. A small rag fire in a corner of the hangar would typically trigger optical fire detectors to release this massive quantity of agent, requiring a costly cleanup for what could have been extinguished easily by hand. The machine vision detector can be set to only sound an alarm for such small, non-growth fires, to alert personnel in the area as well as the fire department. If the fire were to exceed a certain size, or if growth rate became high, the suppression system would be activated. More advanced systems could have directional nozzles for a localized response, avoiding unnecessary cleanup and getting more agent on the actual fire.

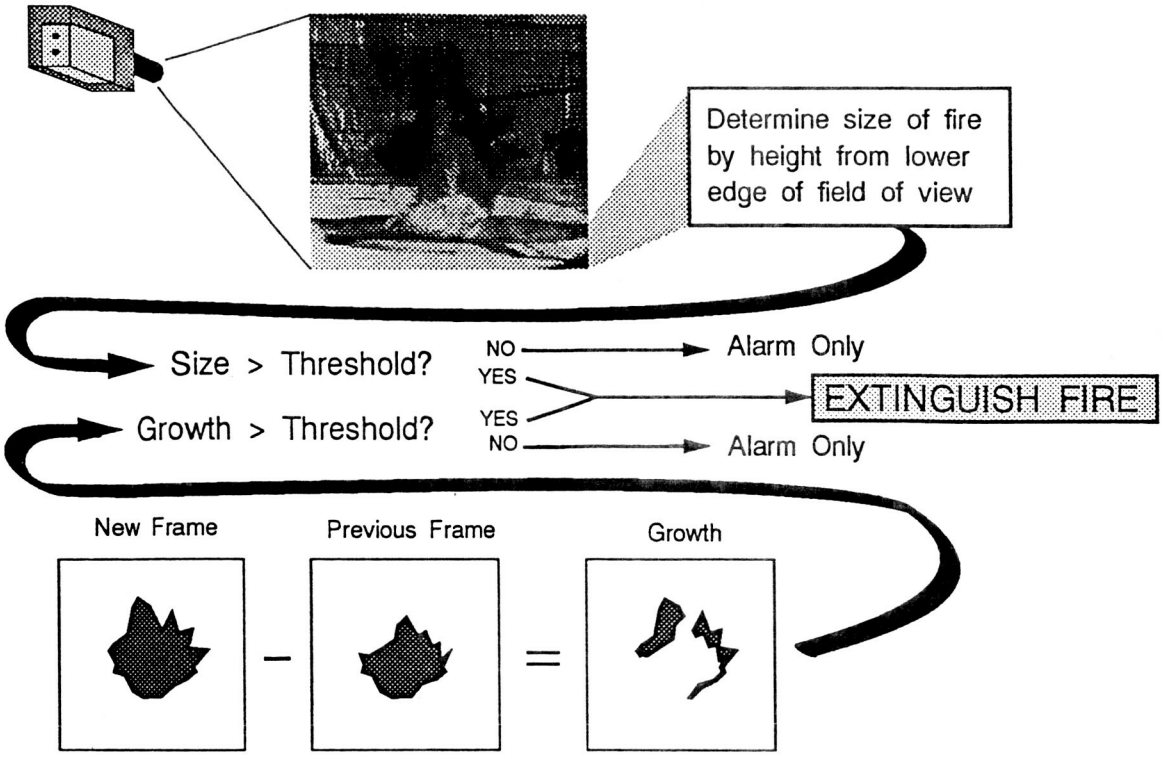


Figure 4. Fire Suppression Decision.

COST AND AVAILABILITY

The hardware components of the system, including the video camera, image processing hardware, and microprocessor, are all available "off the shelf." This drives down cost and technical risk. Current cost of the components is about $2500 and is expected to drop, following the general trend of the small computer industry. New components have become available even during the development process. A new image processing board is being incorporated which will eliminate the need for a separate controlling microprocessor, reducing parts count and cost. The final prototype will consist of the camera, power supply, and one or two circuit boards containing a microprocessor, memory, and all the "rules" for fire detection and decision making. Time for this prototype to make a fire/no fire decision is 1/10 second. Unlike optical fire detectors, speed of detection is unrelated to accuracy; faster times can be achieved, if necessary, through use of a faster microprocessor.

POTENTIAL NASA/SPACE INDUSTRY APPLICATIONS

Machine vision fire detection "knows" what a fire looks like through algorithms embedded in programmable hardware. Because the criteria used in these algorithms is precisely known, the algorithms can be updated to take into account additional threats or false alarm sources tailored for a particular environment. One example, especially applicable to construction and maintenance environments, is to account for luminous sparks, such as from welding. A spark will move rapidly from place to place, unlike a fire, and will have a particular shape. Visual characteristics such as these can be more precisely defined than intensity of radiation sources, which optical fire detectors rely on. In fact, the algorithms can be programmed to identify any visible object with sufficient contrast. For example, vaporized fuel from inadvertent leaks or releases often produces a visible "cloud." Instead of slow-reacting sampling detectors or line-of-sight sensing with restricted coverage, MV detection could be programmed to sense the visible vapor.

Numerous NASA facilities use single-band UV or are scheduled to upgrade to more advanced UV/IR detectors, including shuttle and space station processing bays in the Vehicle Assembly Building, the payload changeout room and transfer arm at the launch pad, and fuel storage and handling facilities. The known limitations of these detectors drive up cost and reduce utility. For example, UV/IR detectors are limited in range because the method depends on the magnitude of emitted energy. The high bays in the VAB are vast spaces over 500 feet high and 400 feet on each side. The UV/IR detectors used are calibrated to detect a 1 square foot fire at 45 feet; thus, many detectors are required to cover this area, at a correspondingly high cost. Machine vision detectors using high-density CCD sensors on the market would have over four times the range. Furthermore, only one computer/processor is needed for up to six cameras, decreasing cost. MV detectors are proving effective where visible flames are involved; near infrared capability would enable the system to detect otherwise invisible hydrazine and hydrogen fires.

For in-flight applications, Machine Vision is a lightweight, reliable alternative. The space station, with its considerable inhabited spaces, will especially require automatic fire detection. One processor could cover a large area, with fiber optics feeding visual information from computer cabinets, equipment enclosures, and other confined areas. A similar concept is scheduled to be developed by the Air Force to protect engine and other internal compartments on the F-22 fighter.

MV detection is a far superior alternative to the single-band fire detection sensors now in common use and has considerable advantages over even the most recent UV/IR detectors. The system is currently in its second prototype stage and has undergone periodic field tests against actual fires and false alarm sources. At the end of 1992, final prototypes will undergo full scale validation. Performance will be evaluated against UV/IR detection for incorporation into a major Air Force upgrade of hangar fire protection systems. The technology will also be applied to development of portable, self-contained fire detection and suppression systems and numerous other applications.

BIBLIOGRAPHY

Goedeke, A.D., Drda, B., Healey, G., Viglione, S., Gross, H.G., Machine Vision Fire Detection System (MVFDS), ESL-TR-91-02, Civil Engineering Laboratory, Air Force Civil Engineering Support Agency, Tyndall Air Force Base, Florida, April 1991.

Session R5: MOBILE ROBOTICS

Session Chair: Dr. Charles Price

REMOTE DRIVING WITH REDUCED BANDWIDTH COMMUNICATION*

Frederick W. DePiero, Timothy E. Noell, and Timothy F. Gee
Oak Ridge National Laboratory
Robotics & Process Systems Division
P. O. Box 2008
Oak Ridge, Tennessee 37831

Presented at the

Sixth Annual Space Operation, Application, and Research Symposium (SOAR '92)
Sponsored by the NASA Johnson Space Center
Houston, Texas

August 6, 1992

*Research sponsored by the U.S. Army Human Engineering Laboratory and managed by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy under contract DE-AC05-84OR21400.

REMOTE DRIVING WITH REDUCED BANDWIDTH COMMUNICATION*

Frederick W. DePiero, Timothy E. Noell, and Timothy F. Gee
Oak Ridge National Laboratory
Robotics & Process Systems Division
P. O. Box 2008
Oak Ridge, Tennessee 37831

ABSTRACT

Oak Ridge National Laboratory has developed a real-time video transmission system for low-bandwidth remote operations. The system supports both continuous transmission of video for remote driving and progressive transmission of still images. Inherent in the system design is a spatiotemporal limitation to the effects of channel errors. The average data rate of the system is 64,000 bits/s, a compression of approximately 1000:1 for the black and white National Television Standard Code video. The image quality of the transmissions is maintained at a level that supports teleoperation of a high-mobility multipurpose wheeled vehicle at speeds up to 15 mph on a moguled dirt track. Video compression is achieved by using Laplacian image pyramids and a combination of classical techniques. Certain subbands of the image pyramid are transmitted by using interframe differencing with a periodic refresh to aid in bandwidth reduction. Images are also foveated to concentrate image detail in a steerable region. The system supports dynamic video quality adjustments between frame rate, image detail, and foveation rate. A typical configuration for the system used during driving has a frame rate of ~ 4 Hz, a compression per frame of ~ 125:1, and a resulting latency of < 1s.

INTRODUCTION

The use of untethered teleoperated vehicles for many remote operations is greatly limited because of a need to use low-bandwidth communication links. Vehicle control over a low-bandwidth channel is a necessity for tactical operations which, for example, require a low signature. Low-bandwidth channels are also encountered in underwater operations and in space applications. The most notable difficulty in using low-bandwidth channels is the problem of video transmissions from the teleoperated vehicle back to the driver's station. Given the availability of a low-bandwidth video transmission system, tactical remote operations such as reconnaissance, surveillance, target acquisition, and convoys, for example, would all become much more feasible. Oak Ridge National Laboratory (ORNL) has developed a real-time video transmission system for these types of low-bandwidth remote operations. The system supports both continuous transmission of video for remote driving and progressive transmission of still images.

The difficulty arising in the transmission of video is its extremely high data rate. Standard black and white video requires 60 M bits per second (bps). State-of-the-art tactical communication links, for example, support data rates in the range of 16 to 64 Kbps. Hydrophone data rates are lower still. A minimum factor of 1000 in video data rate reduction is required for remote driving via these types of

*Research sponsored by the U.S. Army Human Engineering Laboratory and managed by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy under contract DE-AC05-84Or21400.

low-bandwidth channels. Additional challenges exist to the problem of low-bandwidth remote driving beyond the high compression requirement. Driving experience using the ORNL system has demonstrated the significance of latency (image age) on driver performance. Both a low and a constant value of latency is very important. Image latency is affected by the duration of the encoding and decoding processes and by the data rate of the communication channel. Hence, compression techniques that possess deterministic processing times are more applicable for remote driving. The latency requirement together with the high compression rates makes it extremely unlikely that lossless compression techniques will ever be able to provide the required system performance [1]. In light of the fact that low-bandwidth remote driving will suffer from degraded imagery, several human factors studies have examined driver performance under these types of conditions [2][3][4].

Even lossy schemes with a high compression per video frame cannot meet the 1000:1 requirement alone [5]. Some reduction in frame rate is also necessary. Another facet of ORNL's development focused on an image-simulation technique that smoothed the interframe discontinuities associated with a reduced frame rate. These discontinuities are, of course, even more pronounced when the vehicle is driven on rough terrain.

APPROACH

The ORNL system is a hybrid version of the Laplacian pyramid approach to image compression [6]. This method decomposes an image into a set of subimages, each containing a separate spatial frequency band. By stacking the subimages vertically, the shape of a pyramid can be formed and, hence, the term 'image pyramid.' The motivation for this type of approach has its foundation in studies of the human visual system [7][8]. These studies demonstrated the significance of edge information to visual sensing. The studies also revealed a reduced sensitivity to gray-level errors that are present at edges. These results imply that edges should remain present in an image but permit them to possibly contain errors in their intensity values. Pyramidal methods of compression provide this sort of highly selective image degradation.

The block Discrete Cosine Transform (DCT) is another approach that has gained popularity in applications requiring high compression rates [9][10][11]. However, when operated at high compression rates, it suffers from the problem of producing noticeable artifacts at the DCT block boundaries. Block artifacts are not produced by pyramidal methods.

Image pyramids have also been used for scene analysis [12][13][14]. These types of analyses were not part of the ORNL remote-driving project. However, by adopting a compression scheme based on a similar type of image decomposition, results of this project provide an opportunity for a synergistic combination of an analysis plus compression system. An analysis system that could detect nearby obstacles, for example, would be of great benefit to a remote driving system that suffers from degraded imagery.

The hybrid aspect of the ORNL video compression system stems from several extensions to the Laplacian method that have been employed. The overall system uses a combination of several classical compression techniques and image foveation. A foveated image has reduced detail in the peripheral areas. This process mimics the structure of the human eye by placing a region of highest image quality at the center of the operator's field of view. This technique reduces bandwidth while still providing the driver with a feel for the terrain that is passing. The foveal center can be moved by the operator in realtime to adjust to the changing requirements of driving or for some other dynamic aspect of the remote operation such as when a vehicle enters a surveillance mode.

In addition to steering the foveal center, the ORNL system addresses the problem of dynamic adjustments in a more general sense. Five different preprogrammed video-quality settings are provided. These allow the operator to use the available bandwidth in a manner as effective as possible, given changing needs of the remote operation. Trade-offs can be made between image detail and frame rate or between the size of the foveal area and its rate of peripheral degradation, for example.

COMPRESSION ALGORITHM

The first step in producing a Laplacian image pyramid is to create another pyramid known as the Gaussian pyramid. It is generated by recursively applying a low-pass kernel to an image. The low-pass kernel used approximates a two-dimensional Gaussian function [6]. It has a normalized cutoff frequency of $\pi/2.0$ and produces a result that is subsampled by a factor of two in each direction. In this manner, each subsequent subimage, or layer, of the Gaussian pyramid is reduced in area by a factor of four from the previous layer. The ORNL system uses four pyramid layers.

The Laplacian pyramid is formed from the Gaussian pyramid by subtracting adjacent layers. To subtract two layers, the smaller one is expanded in area by a factor of four and then subtracted from its adjacent higher frequency layer. Each layer of the Laplacian pyramid contains a separate spatial frequency band of the original image. Layers of the Laplacian pyramid are referred to as 'subbands' because of this frequency decomposition. Expansion of the Gaussian layers was achieved via pixel replication followed by the application of a 2 x 2 averaging kernel. This method differs from Burt's [6], which used the Gaussian kernel both for expansion and for the recursive low-pass filtering. Slightly higher compression ratios were achieved in the ORNL system by switching to the 2 x 2 averaging kernel for expansion.

Once the Laplacian pyramid has been constructed, a uniform quantizer is applied to each subband. Each subband received a different degree of quantization to take advantage of the varying degree of visual sensitivity to errors in different spatial frequency bands [7][8]. The quantized subbands are then foveated by simply clipping the contents of each band that resides outside a rectangular region (see Figure 1). The centers of each foveal rectangle is collocated in the final image, and the size of each rectangle is determined by the desired rate of foveal degradation. The foveated bands are positioned to produce a gradual shift in image quality from the foveal center out towards the periphery.

To take advantage of the temporal correlation of images, experiments were made in differencing peer subbands in subsequent images. These experiments were designed to determine which bands should be processed in this manner and then to examine the effect of the duration of the differencing process. The process incorporated a periodic refresh to ensure a temporal limitation to channel errors. Taped driving imagery was used for input. The original vehicle transporting the camera had a speed of 10 mph. During the experiments, the compression system ran at a frame rate ranging from 3 to 5 Hz. Under these conditions, the lowest frequency band yielded a 15 to 20% improvement, and the second lowest showed a 5 to 10% improvement. Each varied with the scene content encountered on the tape. The two higher frequency bands did not yield any improvement in the cases examined. Most likely because these bands experienced substantial interframe differences with the vehicle speed and frame rate examined so that a net gain was not realized. The refresh period for both of the lower frequency bands was selected to be four frames. The realized compression did not substantially increase with longer refresh periods. The four-frame interval was chosen as a compromise so that the temporal duration of a channel error was limited to 1s.

Figure 1. An original image (upper left), a compressed image (upper right), and a Laplacian pyramid which has been quantized and foveated.

Classical lossless techniques were employed in two stages to compress the quantized and foveated subbands into a one-dimensional bit stream. First, a zero-run-length coding technique [15] was used to process each row of a band. This process replaces a sequence of zero values with a single symbol indicating the length of the zero run. Nonzero pixels remain unchanged in this operation. Some runs of nonzero pixels did occur in the subbands, but the vast majority of runs consisted of zero pixels. To simplify the coder, it was decided to restrict the formation of runs to be those of zero values only.

The second stage of lossless processing used a Huffman coder [15] to assign a variable-length code word to each zero run and to each nonzero pixel. The code book for the Huffman coder was generated by using statistics gathered from driving imagery. The ORNL system provided the capability of on-line code book generation. An operator could specify starting and stopping times for the accumulation of image statistics while driving. In this way, code books could be tuned in the field for a given terrain. Once generated on board the vehicle, the code book was transmitted (in a lossless mode) to the decoder so that operation could begin on each side of the system using the new code books. Each subband's image statistics varied because of the different quantizers used. Different code books had to be used in each of these cases and when subbands were refreshed rather than differenced. The performance of the Huffman code books dropped by as much as 25% because of variations in scene content.

Using the above two classical techniques yielded a compression slightly higher than the zeroth-order entropy [15] in the higher subbands. An entropy measure is commonly used to determine the theoretically highest compression possible. Note that the calculation assumes a completely random arrangement of symbols in the data set. The quantized bands are far from random. The dominant (nonzero) components in each layer are typically associated with edges in the input image, so layers tend to contain large expanses of zero pixels with small clusters of nonzero values. Hence, the zeroth-order entropy is not a completely accurate metric for the Laplacian subbands.

Each subband was transmitted in a separate packet. Upon reception at the decompression unit, the bit stream was decoded and the image data was painted into frame buffers. The process of collapsing a Laplacian pyramid to recreate an image requires recursive steps of expansion and addition [6]. Several options for scheduling the time to collapse pyramids were explored. Nominally, a pyramid collapse could occur after all the subbands have been accumulated on the decompression side. Another possibility is to collapse the pyramid after each band is received. In the latter approach, final images contain a temporally skewed set of subbands.

A simulation scheme was investigated that employed temporal skewing. Bands were transmitted in order from highest frequency to lowest. A pyramid collapse following the arrival of each new band produced a gradual transition in the image from old to new. The edges lead the change in the image to continually provide the operator with a sense of the changing scene conditions. This technique yielded a pseudoincrease in the frame rate seen by the vehicle operator. Similar concepts have been applied to the area of improved-definition television by using temporal interpolation between subbands of subsequent images. Interpolative techniques are well suited for open-loop systems. Teleoperated systems could benefit from the improved image quality of interpolative approaches but cannot tolerate the accompanying increased latency. The logic behind the simulation approach studied here was to take advantage of subbands immediately upon reception. The use of temporally skewed subbands was considered to be a good starting point for addressing the needs for image simulation in a closed-loop system. Unfortunately, the technique had to be disengaged in the field. The large changes in imagery produced when the vehicle traversed moguls often resulted in noticeable residual artifacts. It became apparent that a more extrapolative technique is required. Future work will address the smooth extrapolation problem.

Another use for temporally skewed bands arises in an opportunity to robustly handle channel dropouts. The packet transmissions of each subband tended to perform in an "all or nothing" fashion. Packets were identified by frequency band and by frame number. If any subbands were dropped, the collapse heuristic was capable of substituting the old version of the same band and collapsing to form a new image. Hence, to be useful, an entire set of bands was not required.

Several close relatives to the Laplacian pyramid have been studied for image compression [16][17][18]. The motivation for choosing the Laplacian over these other methods is due to the property of temporal skewing discussed above. The most notable contender to the Laplacian approach is the Quadrature Mirror Filter (QMF). The QMF kernel produces a somewhat more compact decomposition of images than does the Laplacian and, consequently, has been more closely examined in recent years. Some debate has arisen over the merits of the Laplacian versus the QMF kernel. Vetterli contends, for example, that the Laplacian is a superior choice overall because of improved results in the area of motion-adaptive compression [19]. The Laplacian approach was chosen for use with the ORNL system for closely related reasons. The Laplacian has a tolerance to misregistration of subbands during the collapse process. The high-pass version of the QMF kernel produces bands that must be precisely aligned prior to collapsing to prevent noticeable artifacts. Given that driving imagery is typically always varying and because of an interest in temporally skewing subbands, the Laplacian approach to pyramid generation was chosen for the ORNL system.

SYSTEM ARCHITECTURE

The final configuration of the ORNL system consists of a two Versa Module European (VME) racks, one for compression and one for decompression. The compression rack is mounted on board a high-mobility multipurpose wheeled vehicle (HMMWV) outfitted for teleoperation. The decompression rack is mounted in an environmental enclosure adjacent to a VME-based Sparc station. An operator interface runs on the Sparc, providing vehicle control functions. The driving station and teleoperated HMMWV were developed by Harry Diamond Laboratories (HDL). The ORNL compression system provides video support for an Automatic Target Acquisition (ATA) system on board the HMMWV. The compression system transmits seven high-resolution still images at the start of targeting operations. As the ATA system acquires targets, the compression system transmits small rectangular portions of images containing the tracked targets. These were typically ~40 x 60 pixel in size. At the operator station, the decompression system pastes targets at appropriate locations within the frame buffers. The contents of the tracking buffers are displayed on the operator control station. A socket-based custom protocol is used to communicate between the ORNL compression system, the operator control station, and the ATA system.

Each rack contains three single-board computers, Datacube image processing hardware, and memory cards. The first processor in the compression rack is responsible for controlling the Datacube equipment and for determining the image capture rate. The second processor performs the zero-run-length and Huffman coding. The third interfaces to the radio. The processors in the decompression rack also form a pipeline for images and perform symmetrical functions.

The packet radios use a spread-spectrum type of modulation and operate in the 902 to 928-MHz band. The units provide a wireless Ethernet bridge between the VME systems. The low-bandwidth communication channel is emulated by using a real-time clock to maintain a data rate at 64 Kbps. The average rate was also monitored at the receiving unit for verification. Fluxuations down to ~ 55 Kbps commonly seen were due to competing (RF) traffic. It was necessary to adjust the physical packet size of the transmitted data to improve the system's ability to coexist with other nearby transmitters in the same RF band. The physical packet size was adjusted by modifying the operating systems' Ethernet driver.

The ORNL system is capable of using either protocol with the Ethernet bridge. TCP sockets guarantee faithful delivery of all packets, in order, and will retry indefinitely to provide such. UDP sockets do not provide a similar guarantee. This flexibility provides the option to avoid exhaustively attempting to retransmit old subbands. In the event of a dropout, the system simply begins the transmission of the next new band. Given the tolerances of the Laplacian pyramid described above, bands can be dropped without severe consequences. At the demonstration of the system, it was operated in a TCP mode. Hence, the effect of channel errors on video quality was not a visual blemish. Rather, it was an increased delay in transmission for that band. It is believed that the design aspects affecting channel noise tolerance that were made part of the system were a worthwhile development effort, although they were not fully exercised at the first demonstration. It is anticipated that these aspects of the system will come to fruition with future versions of the system.

In addition to the two-rack compression system, a one-rack simulation system has also been developed. The simulator has been delivered to the U.S. Army's Human Engineering Laboratory (HEL) for human factors studies on remote driving. The simulator is capable of producing the same degraded imagery and of emulating the latency present in the two-rack version of the system. Future work at ORNL will incorporate the recommendations indicated by HEL's studies.

CONCLUSIONS

The ORNL video compression system was demonstrated in April 1992. The system supports teleoperation of an HMMWV at speeds up to 15 mph on a moguled dirt track. During driving tests, the compression per frame ranged from ~ 105:1 to 145:1, depending on scene content. The frame rate varied as a function of the realizable compression, ranging from 3 to 6 Hz. Latency of the system was determined to be ~ 1s. Future work in this area will address improvements to the compression algorithm, the problem of temporal extrapolation, and the transmission of color images.

ACKNOWLEDGMENTS

The authors thank Mr. Charles M. Shoemaker and Mr. Thomas W. Haduch for programmatic guidance throughout this project. The authors acknowledge the cooperative attitude and excellent development efforts by the members of the Advanced Sensor Systems Division of HDL.

REFERENCES

[1] Hong, T. H., Nashman, M., and Chaconas K. (1988). "Image Compression Technology and Techniques," unpublished manuscript, National Bureau of Standards, Gaithersburg, Md.

[2] McGovern, D. E. (1987). "Experiences in teleoperation of land vehicles" (SAND87-1980C), Albuquerque, N.M.: Sandia National Laboratories.

[3] Shoemaker, C. M. (1990) "Low data rate control of unmanned ground vehicles," *proceedings: Association for Unmanned Vehicle Systems AUVS-90*, Dayton, Ohio.

[4] Miller, D. P. (1987). "Evaluation of vision systems for teleoperated land vehicles," *proceedings IEEE Systems, Man and Cybernetics Conference.*

[5] Herman, M., Chaconas, K., Nashman, M., and Hong, T. H. (1988). "Video compression for remote vehicle driving." *Proceedings SPIE Advances in Intelligent Robotics Systems: Mobile Robots III*, Cambridge, Mass.

[6] Burt, P. J., and Adelson, E. H., "The Laplacian Pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, pp. 532-540, April 1983.

[7] Carlson, C. R., and Cohen, R. W., *Visibility of displayed information*, Office of Naval Research, Technical Report, Contract N00014-74-C-0184, 1978.

[8] Carlson, C. R., and Cohen, R. W., "A simple psychophysical model for predicting the visibility of displayed information," *Proc. Soc. Inform. Display*, pp. 229-246, 1980.

[9] Ahmed, N., Natarajan, T., and Rao, K. R., "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90-93, January 1974.

[10] Polysongsang, A., and Rao, K. R., "DCT/DCPM Processing of NTSC composite video signal," *Proc. Int. Symp. Circuits Syst.*, Tokyo, Japan, July 17-19, 1979, pp. 981-982.

[11] Ligtenberg, A., and Wallace, G., K., The emerging JPEG compression standard for Continuous Tone Images: An Overview," *Proc. Picture Coding Symposium, Cambridge, Mass., March 1990*, pp. 6.1-1.—6.1-4.

[12] Anderson, C. H., Burt, P. J., and van der Wal, G. S., "Change detection and tracking using pyramid transform techniques," *Proc. SPIE Conf. on Intell. Robots and Computer Vision*, Boston, pp. 72-78, 1985.

[13] Bergen, J. R., and Adelson, E. H., "Hierarchical, computationally efficient motion estimation algorithm," *J. Opt. Soc. Am.* A, 4, p. 35, 1987.

[14] Burt, P. J., "Multiresolution techniques for image representation, analysis, and 'smart' transmission," *Proc. SPIE Conf. on Visual Communications and Image Processing*, pp. 2-15, Philadelphia, Pa., November 1989.

[15] Jain, A. K., "Image data compression: A review," *Proceeding of the IEEE*, vol. 69, pp. 349-384, March 1981.

[16] Vetterli, Martin, "Multidimensional sub-band coding: some theory and algorithms," *Signal Processing*, 6(2):97-112, February 1984.

[17] Adelson, E. H., Simoncelli, E., and Hingorani, R., "Orthogonal pyramid transforms for image coding," *Proceedings of SPIE*, October 1987.

[18] Pentland, A., and Horowitz, B., "A practical approach to fractal-based image compression," Technical Note 152, M.I.T. Media Lab Vision and Modeling Group, Boston, Mass., November 1990.

[19] Uz, K. M., Vetterli, M., and LeGall, D. J., "Interpolative multiresolution coding of advanced television with compatible subchannels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 1, March 1991.

PLANETARY ROVER DEVELOPMENTS AT JPL

Mr. Brian Wilcox
Jet Propulsion Laboratory
M/S 107-102
4800 Oak Grove Dr.
Pasadena, CA 91109

Planetary rover research has recently focused on small rovers, which are competent to explore and assist in in-situ analysis of limited areas around a landing site (as opposed to the larger, long-range rovers, which have been assumed to accompany sample return missions). Navigation and mobility concepts of these small rovers are, in some cases, somewhat different than has been assumed in the past. The sensing, computing, communication, and power resources of these missions dictate a rethinking of the "large mission" approach. Recent results and demonstrations along this new direction are described.

REAL-TIME QUALITATIVE REASONING FOR TELEROBOTIC SYSTEMS

Dr. Eancois G. Pin
Oak Ridge National Laboratory
Center for Engineering Systems Advanced Research
P.O. Box 2008
Oak Ridge, TN 37831-6364

This paper discusses the sensor-based telerobotic driving of a car in a-priori unknown environments using "human-like" reasoning schemes implemented on custom-designed VLSI fuzzy inferencing boards. These boards use the Fuzzy Set theoretic framework to allow very vast (30 kHz) processing of full sets of information that are expressed in qualitative form using membership functions. The sensor-based and fuzzy inferencing system has been incorporated on an outdoor test-bed platform to investigate two control modes for driving a car on the basis of very sparse and imprecise range data. In the first mode, the car navigates fully autonomously to a goal specified by the operator, while in the second mode, the system acts as a telerobotic driver's aid providing the driver with linguistic (fuzzy) commands to turn left or right, speed up, slow down, stop, or back up depending on the obstacles perceived by the sensors. Indoor and outdoor experiments with both modes of control are described in which the system uses only three acoustic range (sonar) sensor channels to perceive the environment. Sample results are presented that illustrate the feasibility of developing autonomous navigation modules and robust, safety-enhancing driver's aids for telerobotic systems using the new fuzzy inferencing VLSI hardware and "human-like" reasoning schemes.

R.A.T.L.E.R.

Robotic All-Terrain Lunar Exploration Rover

J.W. Purvis* and P.R. Klarer**
Sandia National Laboratories
Albuquerque, New Mexico

A robotic rover vehicle designed for use in the exploration of the Lunar surface is described. The Robotic All-Terrain Lunar Exploration Rover (R.A.T.L.E.R.) is a four wheeled all-wheel-drive dual-body vehicle. A uniquely simple method of chassis articulation is employed which allows all four wheels to remain in contact with the ground, even while climbing over step-like obstacles as large as 1.3 wheel diameters. Skid steering and modular construction are used to produce a simple, rugged, highly agile mobility chassis with a reduction in the number of parts required when compared to current designs being considered for planetary exploration missions. The design configuration, mobility parameters, and performance of several existing R.A.T.L.E.R. prototypes are discussed.

INTRODUCTION

In 1989 President George Bush called for the establishment of a U.S. Space Exploration Initiative with the goals of returning to the Moon to stay and a manned mission to Mars. Subsequent national studies such as NASA's 90-Day Study[1] and the Synthesis Report[2] have led to significant renewed interest in robotic precursor missions for exploration of the Moon. The recent Lunar Rover/Mobility Systems Workshop[3], conducted by NASA's Exploration Program Office and the Lunar Planetary Institute, proposed two initial missions and established some criteria for Lunar rover platforms.

For Lunar exploration missions lasting one Lunar day or longer, a robotic rover system must combine high agility and efficient thermal management with radiation hardness to assure a high probability of mission success in that extreme operating environment. Low launch mass, high reliability, and robustness of the system are highly desirable characteristics as well, since the vehicle will not be readily accessible for repair or recovery once it has been deployed. Engineers at Sandia National Laboratory have recently demonstrated an innovative concept for a robotic rover vehicle designed for use in the exploration of the Lunar surface. The design configuration, mobility parameters, and performance of several prototypes of this vehicle are discussed below.

---

*Senior Member of Technical Staff, Strategic System Studies Center.
**Senior Technical Associate, Advanced Vehicle Technologies Department.

VEHICLE DESCRIPTION

The Robotic All-Terrain Lunar Exploration Rover (R.A.T.L.E.R.) is a four wheeled all-wheel-drive platform with twin body compartments connected by a hollow central pivot. The general configuration is shown in Figure 1, which is a three-view schematic of the dual-body central-pivot design for which a patent has been applied.

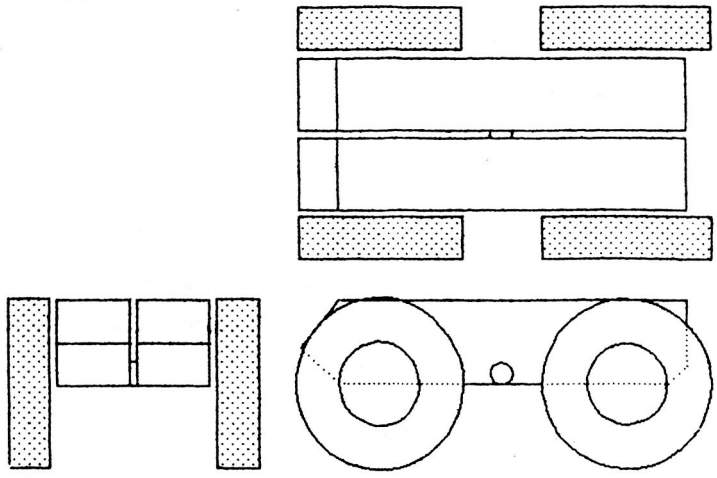


Figure 1. R.A.T.L.E.R. Schematic.

The uniquely simple method of chassis articulation by means of the hollow central pivot is employed between the bodies to allow all four wheels to remain in contact with the ground while traversing uneven terrain. This central pivot, as well as the vehicle center of mass, is located as close to the axle line and the geometric center of the vehicle as possible to ensure maximum stability while climbing over large obstacles. The articulating action of the dual-body central-pivot is illustrated in Figure 2, which is a picture of the first remotely controlled prototype being driven over some large rocks in Death Valley.

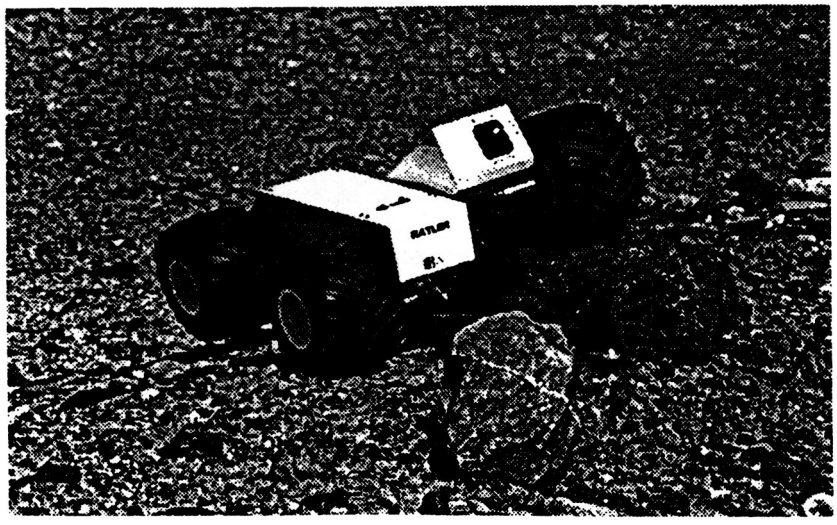


Figure 2. R.A.T.L.E.R. Prototype at Death Valley.

ANALYSIS

Only four situations have been analyzed to date for the R.A.T.L.E.R.: 1) the maximum height vertical step which can be cleared, 2) the optimum wheelbase for a given wheel diameter, 3) an estimate of the optimum stance, and 4) the traction advantage of the dual-body central-pivot design over a conventional four wheeled platform.

Figure 3 shows a schematic of the maximum step height problem. From the geometry, the maximum step height H which can be climbed is

1) $$H_{max} = \sqrt{B^2 - R^2}$$

where B is the wheelbase and R is the wheel radius. The ground clearance is assumed to equal to the wheel radius, and the center of mass is assumed to be centered along the line of axles so that the vehicle will not tip over backwards.

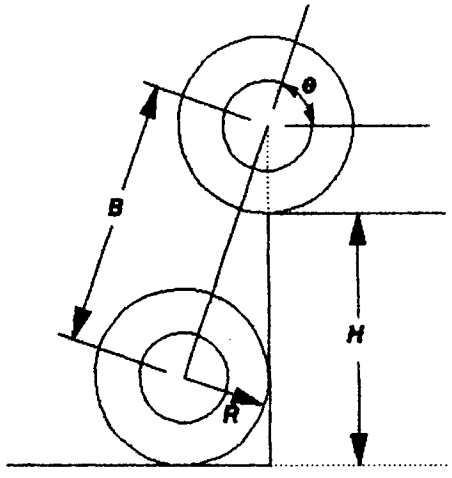


Figure 3. Maximum step clearance geometry.

Constructing a figure similar to Figure 3 but without the rear wheel touching the vertical face of the step, it can be shown that for any angle θ there is a minimum height

2) $$H_{min} = R(1+\tan\theta)$$

such that the vehicle bottom will scrape on all steps with height H if

3) $$H_{min} < H < H_{max}$$

In terms of the geometry

4) $$H = B\sin\theta$$

and therefore no scraping will occur as long as

5) $$B\sin\theta < R(1+\tan\theta)$$

The bottom will just touch the point of the step if the inequality in Equation (5) is instead an equality. Assuming the equality and differentiating with respect to θ, it can be shown that a minimum occurs when

6) $$\sin\theta = \cos\theta$$

The optimum value of B for no bottom scraping is then

7) $$B = 2\sqrt{2}\,R$$

Using the above result in Equation (1), the maximum step height that a vehicle with optimized wheelbase can climb is

8) $$H_{max} = \sqrt{7}\,R$$

which is 1.32 wheel diameters. A similar analysis using Equation (8), assuming hemispherical boulders, and neglecting wheel width, shows that the optimum stance S must be in the range

9) $$S < 3.74\,R$$

A much more complicated analysis of the step problem, which will not be repeated here, shows that when only one side of the vehicle climbs a step, the leverage advantage of the articulating R.A.T.L.E.R. design requires only half as much torque to climb the step as a conventional four wheeled platform. This is intuitively obvious since only "half" of the R.A.T.L.E.R. vehicle is traversing the obstacle. Alternatively, if the two vehicles have equal torque, the R.A.T.L.E.R. design can climb steps which are slicker by almost a factor of two (the coefficient of friction equations are not linear).

PROTOTYPES AND TESTS

The original prototype was a small, unpowered, uncontrolled balsa model about six inches long which was used to verify that the dual-body central-pivot concept would traverse large obstacles and had some advantages over conventional platforms. Several other models were then constructed to investigate conventional steering versus skid steering, body shapes, tethered controls, remote RF controls, and even solar power use.

The first large scale prototype, incorporating the best ideas from all of the early models, is the one shown in Figure 2 which has been dubbed the "White R.A.T.L.E.R.". This model is about 15 inches long with a balsa, mylar, and plastic tubing chassis. The drive system consists of four 7/8ths inch dia by 3 inches long constant speed DC electric motors reclaimed from the DOE weapons program. Skid steering is used. The control system is a model aircraft radio control set coupled to microswitches for motor control and standard servo setups for the internal tilt of the miniature CCD onboard camera. The entire system, including the remote video transmitter, is powered by a series of 9V transistor radio batteries. This prototype has been tested extensively for obstacle climbing abilities, and once in the sands at Death Valley.

Three subsequent prototypes, all using skid steering, have been constructed and are now undergoing extensive testing at Sandia and on the dunes at White Sands National Monument (WSNM). One is an eleven pound aluminum replica of the White R.A.T.L.E.R. powered by a series of 12V gelcells, and with an external pan and tilt miniature CCD camera. The second unit is an 8 inch Pygmy R.A.T.L.E.R. with external pan and tilt CCD camera, miniature video transmitter, and variable speed drive system for the wheels. The last unit is a flat plate body testbed with variable speed drive system, designed so that the stance, ground clearance, pivot height, and pivot limits can be easily changed. All three systems have been tested in damp gypsum sand at WSNM and can climb 18-22 degree slopes. A dry, powdery sand test with the Pygmy R.A.T.L.E.R. showed the potential for climbing even steeper slopes, but further tests are needed to verify this observation.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the many individuals who have directly or indirectly contributed to the R.A.T.L.E.R. project: Kent Biringer, coinventor of the original concept; Adan Delgado, Leon Martine, and Patrick Wing, Sandia summer students who constructed and tested the last three prototypes; George Abbey, of the National Space Council, Dr. Fenton Carey, of the DOE Office of Space, and Gen. Tom Stafford, astronaut, for their enthusiastic encouragement; and finally our many colleagues at NASA, whose comments, constructive criticisms, and support have greatly influenced the R.A.T.L.E.R. development.

REFERENCES

1 NASA, "90-Day Study Data Book," NASA internal report, 1989.

2 Stafford,T. *et al.*, **America at the Threshold: Report of the Synthesis Group on America's Space Exploration Initiative**, U.S. Government Printing Office, 1991.

3 Hoffman, S.J., and Weaver, D.B.,"Results and Proceedings of the Lunar Rover/Mobility Systems Workshop," EXPO-T2-920003-EXPO, NASA, 1992.

Session R6: POTENTIAL FLIGHT EXPERIMENTS

Session Chair: David Lavery

POTENTIAL ROLES FOR EVA AND TELEROBOTICS IN A UNIFIED WORKSITE

Dr. David Akin and **Dr. Russel D. Howard**
University of Maryland
Space Systems Laboratory
Dept. of Aerospace Engineering
College Park, MD 20742

Although telerobotics and extravehicular activity (EVA) are often portrayed as competitive approaches to space operations, ongoing research in the Space Systems Laboratory (SSL) has demonstrated the utility of cooperative roles in an integrated EVA/telerobotic worksite. Working in the neutral buoyancy simulation environment, tests have been performed on interactive roles or EVA subjects and telerobots in structural assembly and satellite servicing tasks. In the most elaborate of these tests to date, EVA subjects were assisted by the SSL's Beam Assembly Teleoperator (BAT) in several servicing tasks planned for Hubble Space Telescope, using the high-fidelity crew training article in the NASA Marshall Neutral Buoyancy Simulator. These tests revealed several shortcomings in the design of BAT for satellite servicing and demonstrated the utility of a free-flying or RMS-mounted telerobot for providing EVA crew assistance. This paper documents the past tests, including the use of free-flying telerobots to effect the rescue of a simulated incapacitated EVA subject, and details planned future efforts in this area, including the testing of a new telerobotic system optimized for the satellite servicing role, the development of dedicated telerobotic devices designed specifically for assisting EVA crew, and conceptual approaches to advanced EVA/telerobotic operations such as the Astronaut Operations Vehicle.

N93-32119

Graphics Simulation and Training Aids for Advanced Teleoperation

Won S. Kim, Paul S. Schenker, and Antal K. Bejczy

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

ABSTRACT

Graphics displays can be of significant aid in accomplishing a teleoperation task throughout all three phases of off-line task analysis and planning, operator training, and on-line operation. In the first phase, graphics displays provide substantial aid to investigate workcell layout, motion planning with collision detection and with possible redundancy resolution, and planning for camera views. In the second phase, graphics displays can serve as very useful tools for introductory training of operators before training them on actual hardware. In the third phase, graphics displays can be used for previewing planned motions and monitoring actual motions in any desired viewing angle, or, when communication time delay prevails, for providing predictive graphics overlay on the actual camera view of the remote site to show the non-time-delayed consequencies of commanded motions in real time. This paper addresses potential space applications of graphics displays in all three operational phases of advanced teleoperation. Possible applications are illustrated with techniques developed and demonstrated in the Advanced Teloperation laboratory at Jet Propulsion Laboratory. The examples described include task analysis and planning of a simulated Solar Maximum Satellite Repair task, a novel force-reflecting teleoperation simulator for operator training, and preview and predictive displays for on-line operations.

I. INTRODUCTION

Advances in computer graphics technologies enable the design, development, and use of high-fidelity graphics displays for very efficient operation aid in space telerobotics. Advanced graphics techniques [10] can be used to achieve increased reliability in all three phases of space telerobotic operations: in off-line task analysis and planning, in operator training, and in on-line task execution. This paper addresses potential use of graphics displays in all three phases of space telerobotics flight experiments. All three applications areas are described and illustrated by examples implemented at the Advanced Teleoperation Project [3], [16] at the Jet Propulsion Laboratory.

II. TASK ANALYSIS AND PLANNING DISPLAYS

Graphics displays can provide substantial aid in off-line task analysis and planning, for example, to investigate workcell layout, motion planning with collision detection and with possible redundancy resolution, and planning for camera images. Graphics displays are used for task analysis and planning of Solar Maximum Repair Task. The Solar Maximum Repair Mission (SMRM) [4] was successfully completed by two astronauts through a 7-hour extra vehicular activity (EVA) in 1984. In this mission, the Solar Maximum Mission (SMM) satellite was captured and berthed in the Space Shuttle cargo bay by using the Shuttle Remote Manipulator System (RMS), and then three tasks of replacing Modular Attitude Control System (MACS), installing a protective cover, and replacing the Main Electronics Box (MEB) of an instrument were performed prior to the deployment of the repaired satellite. The most difficult task among the three was the MEB repair. Central Research Laboratories (CRL) demonstrated in 1987 that the MEB repair task could be performed by teleoperation using a 7-dof force-reflecting master/slave manipulator system [1]. The same MEB repair task is planned to be demonstrated in the Advanced Teleoperation Laboratory (ATOP) by using a dual-arm teleoperation system equipped with recent advanced control and graphics display techniques. Two 8-dof redundant robot arms from AAI (American Armament Inc.) has just been installed, replacing the two existing 6-dof PUMA arms, to increase the reach volume and dexterity.

The IGRIP (Interactive Graphics Robot Instruction Program) software package from Deneb Robotics [5] is used in our initial off-line task analysis/planning of the simulated SMSR task. The package provides an excellent operator-interactive graphics simulation environment with advanced features for CAD-based model building, workcell layout, collision detection, path designation, and motion simulation.

The workcell of the simulated SMSR task shown in a graphics display of Fig. 1 consists of two 8-dof AAI robot arms, a partial SMM satellite mockup, two "smart" hands (end effectors), a raised tile floor, and a screw driver tool. Other workcell elements include camera gantry frame and

various other end effector tools such as a tape cutter to cut Kapton tapes and a diagonal cutting plier to cut tie wraps. By using CAD-based object model building functions of IGRIP, each device in the workcell was built by first creating the individual parts of the device and then putting them together with appropriate definitions of kinematic attributes. All the graphics models were built by using actual dimensions measured. After all the devices were built, these devices were laid out in the workcell by using workcell layout functions of IGRIP. Each device is allowed to be moved in any position and orientation with a mouse. In order to determine the desirable mounting locations of the robots and the satellite mockup, reach envelopes of a robot were overlaid on the workcell display graphics for various task conditions, where each device was allowed to be moved to satisfy the reach envelope constraints.

The AAI arm which has just been installed for the SMSR demonstration is an 8-dof redundant manipulator with 8 rotational joints connected serially. The axes of the first 3 joints intersect with each other at the shoulder point. Joint 4 specifies the elbow bending angle, and the first 4 joints (joint 1 through 4) determine the wrist position. The remaining 4 joints constitute a wrist, and their axes intersect with each other at the wrist point. The lengths of the upper arm (from shoulder to elbow) and the forearm (from elbow to wrist) are 27.407 in. and 21.930 in., respectively. By considering the geometry of the AAI arm, we can easily observe that the wrist reach volume which is the set of points reachable by the robot wrist point is a relatively thin spherical shell of of thickness 17.348 in. with inner and outer radii 31.989 in. and 49.337 in., respectively.

An example of the reach envelope analysis is shown in Fig. 2 to determine the opening angle of the MEB panel. Removal of electric connector screws from the MEB requires that the screw driver held by the right robot hand be able to reach all the connector screws at the right angle to the MEB panel. The total length from the wrist to the screw driver tip is 26.125 in. (screw driver tool length = 7.5 in.), and the reach envelope of the screw driver for the perpendicular orientation to the panel is obtained by translating the wrist reach envelope of Fig. 2 by 26.165 in. towards the panel. When the panel is 100° opened, some of the connector screws near the hinge assembly cannot be reached by the screw driver at the right angle (Fig. 2a). A rather long distance from the wrist to the tool tip severely restricts acceptable opening angles of the panel. Further careful reach envelope analysis indicates that when the panel is 115° opened, the screw driver can reach all the connector screws at the right angle to the panel (Fig. 2b).

An inverse kinematic routine developed for the 8-dof redundant AAI arm [14] was incorporated into each of the two graphically simulated robot arms to allow cartesian robot control. The inverse kinematic algorithm employed is a simplistic approach of fixing 2 joints and using only 6 joints at a time. At present joint 3 and 5 are chosen as fixed joints and used as redundancy control parameters. In the normal operating region of the SMSR task, the joint 3 value is

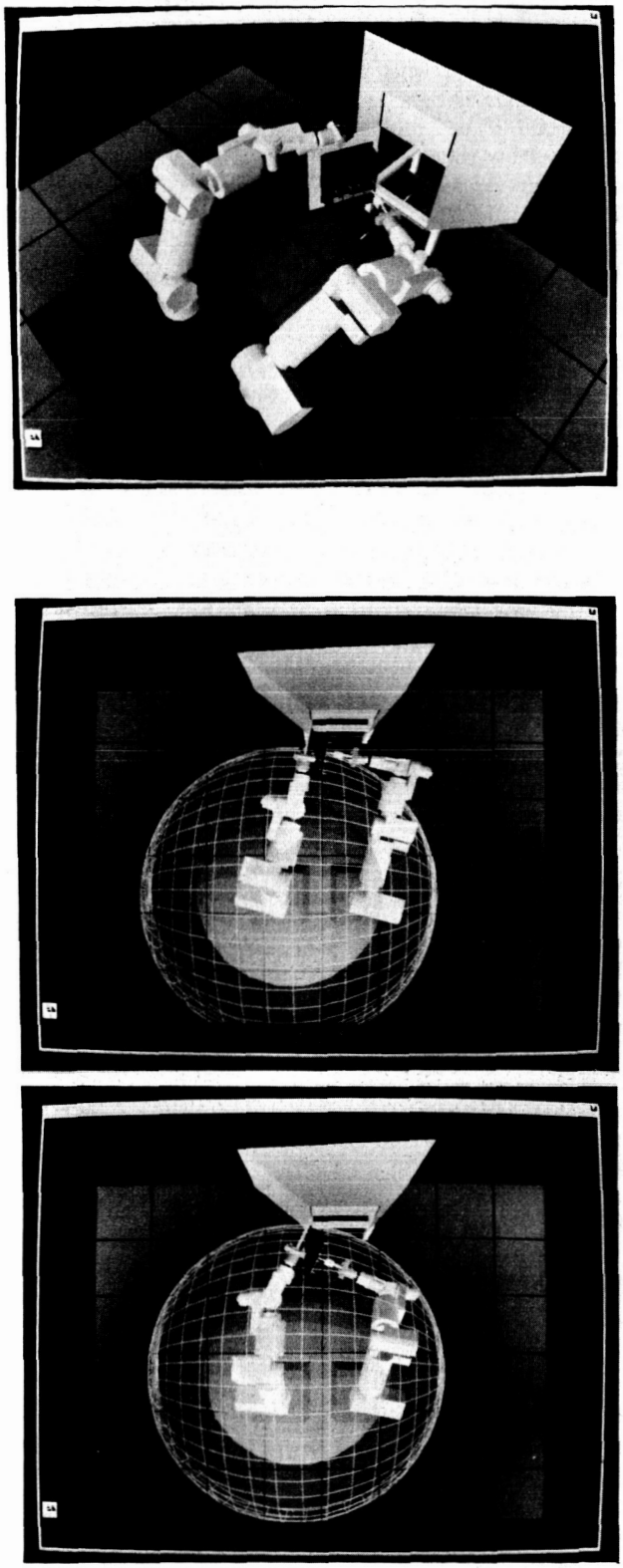


Fig. 2. The MEB panel of the satellite mockup should be inside the reach envelope of the right robot hand for folding and unfolding thermal blankets (upper) and inside the reach envelope of the tape cutter held by the right robot hand for Kapton tape cutting (lower).

closely related to the elbow rotation about the axis connecting the shoulder and wrist points. However, when the desired wrist position is right above the shoulder point, joint 3 must be 0° or 180° and in this case only joint 1 is closely related to the elbow rotation angle. The cartesian motion of each robot was verified graphically with the "T-jog" function of IGRIP. In the "T-jog" mode, the end effector frame of the robot follows the coordinate frame of a selected tag point which can be controlled with a mouse or a keyboard. The graphically simulated arms can also be controlled with hand controllers using the inverse kinematic routine residing in the real-time robot control system. This helps to check the inverse kinematic routine implemented on the real-time system.

In Fig 3, the joint 3 values of the left and right arms are inadequately set to 105° and −60°, respectively. As a result, the elbows of the two arms begin to collide with each other, as we move the right arm further away from the satellite mockup keeping the screw driver tip on the MEB surface and the orientation at the right angle to the panel. Fig. 3 shows a configuration right before collision. When the two arms collide, the collision is immediately detected by IGRIP, and the parts in collision get highlighted in red. The two arms containing the colliding parts get highlighted in cyan, and the rest of the devices in the workcell turn blue. Of course, an adequate setting of the elbow rotation angles can avoid collisions as in Fig. 1.

It cannot be overemphasized that good camera viewing conditions are essential to perform teleoperation missions successfully. Simulated graphics displays of camera views can be used for sensor planning to determine desirable camera locations and zoom settings for each task segment by using viewing criteria such as visibility, resolution, field of view, and lighting. Continuous motion simulations can be performed by using programs written in GSL (graphics simulation language) and CLI (command line interpreter) supported by IGRIP. Collisions between devices in the workcell can be detected during simulation. A final verified planned task sequence and the motion simulation for each task segment can be used later for preview display during the on-line task execution.

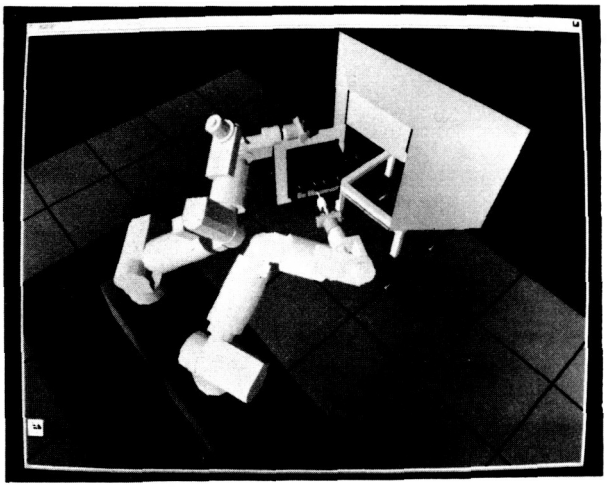


Fig. 3. Redundance resolution with the joint 3 value. An inadequate setting of the elbow rotation angle can cause collisions between the two arms.

III. OPERATOR TRAINING DISPLAYS

Graphics displays can also serve as an introductory training tool for operators. Teleoperation in general demands considerable training, and robots can be damaged during the initial stages of the training. Prior to training with actual robots, a telerobot simulator can be used during the initial training. Introductory training with a simulator can save time and cost for space crew training.

Recently we have developed a force-reflecting teleoperation simulator/trainer [7], [10], [12] as a possible computer-aided teleoperation training system (Fig. 4). A novel feature of this simulator is that the operator actually feels virtual contact forces and torques of a compliantly controlled robot hand through a force reflecting hand controller during the execution of the simulated peg-in-hole task. The simulator allows the user to specify force reflection gains and the stiffness (compliance) values of the manipulator hand for both the three translational and the three rotational axes in Cartesian space. The location of the compliance center can also be specified, although initially it is assumed to be at the grasp center of the manipulator hand.

A peg-in-hole task is used in our simulated teleoperation trainer as a generic teleoperation task. An in-depth quasi-static analysis of a two-dimensional peg-in-hole task has been reported earlier [17], but the two-dimensional model is not sufficient to be utilized in a teleoperation trainer. This two dimensional analysis is thus extended to a three-dimensional peg-in-hole task [10], [12], so that the analysis can be used in our simulated teleoperation trainer. In our three-dimensional peg-in-hole task simulation, both the hole and the peg are assumed to be cylindrical with radii of R and r, respectively (Fig. 5). Throughout the analysis, we also assume that the clearance of the hole is small, and thus the angle between the peg and hole axes is assumed to be sufficiently small, allowing small angle approximation. In general, the peg rotates and translates during execution of the peg-in-hole task, accommodating itself to the hole structure by correcting lateral and angular errors of the operator-commanded peg position and orientation. In order to have finite contact forces and torques, both lateral and angular compliance must be provided for the system. In our simulation, the hole and its support structure are assumed to be rigid with infinite stiffness, while the robot hand holding the peg is compliant for all three cartesian translational axes and also for all cartesian rotational axes (Fig. 5). We further assume that the compliance center is located at a distance L from the tip of the peg with three lateral springs k_x, k_y and k_z and three angular springs k_{mx}, k_{my}, k_{mz}. Both the operator-commanded and the actual positions of the peg are described by the position of the compliance center. No friction is assumed throughout the analysis.

For a given operator-commanded peg position, the actual peg position after compliant accommodation can be different depending upon the current state of the peg of whether the peg is currently in the hole or not. When the peg is not currently in the hole (peg-not-in-hole state), three conditions are possible: i) the peg is not in contact with the wall (no-contact condition), ii) the peg is in contact with the wall (peg-on-wall condition), and iii) the peg is in contact with the entrance of the hole. When the peg is in the hole (peg-in-hole state), four conditions are possible: i) no-contact, ii) peg-side one-point contact, iii) peg-tip one-point contact, and iv) two-point contact. In our initial development of computing virtual kinesthetic contact forces and torques, a rough approximation was used during the initial insertion transition from the peg-not-in-hole state to the peg-in-hole state. Detailed computational procedures can be found in [10], [12]. A more generalized method of computing contact forces and torques based on a general collision detection algorithm is under consideration.

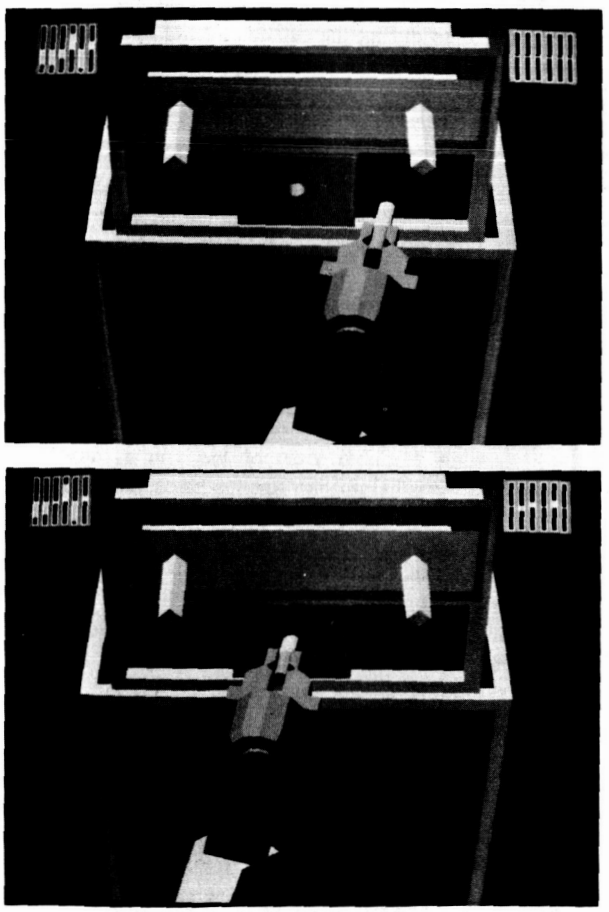


Fig. 4. Force-reflecting teleoperation training displays before contact (upper) and during insertion (lower). Contact forces and torques are computed and reflected to the force reflecting hand controller in real-time. They are also displayed on the upper left corner of the screen, while the current joint angles appear on the upper right corner.

A high fidelity real time graphics simulation of the peg-in-hole task with a PUMA arm and a generic task board has been accomplished by using a Silicon Graphics IRIS-4D/310 VGX workstation, which is very fast both in computation and in graphics rendering with hardware-supported hidden surface removal and lighting. When no contact computations are involved, the update rate of our peg-in-hole graphics simulation is as fast as the display refresh rate: 60 frames/s for workstation display and 30 frames/s for NTSC video monitor display. When force/torque computations are involved due to contact, the worst update rate is about 16 frames/s. The 6-dof hand controller motion commanded by the human operator is sent to the graphics simulation display through a serial I/O line at an about 30 Hz data update rate. Virtual contact forces and torques are computed in real time and fed back to the hand controller through the serial I/O line at an about 30 Hz data update rate. The round-trip time delay of our force-reflecting simulator system from the operator position command to the force reflection to the operator is about 30 to 80 ms.

Testings with the developed peg-in-hole task simulator/trainer indicate that appropriate compliance values are essential to achieve stable force-reflecting teleoperation in performing the simulated peg-in-hole task. As the compliance values of the simulated robot hand becomes smaller, the operator must hold the force-reflecting hand controller more firmly to maintain the stability of teleoperation. In the current implementation, robot servo system dynamics are not included.

So far we have described graphics displays for off-line task analysis/planning and simulated training. Graphics displays can also provide effective operator aid during the on-line operation. Two application examples of preview and predictive displays are described the next two sections.

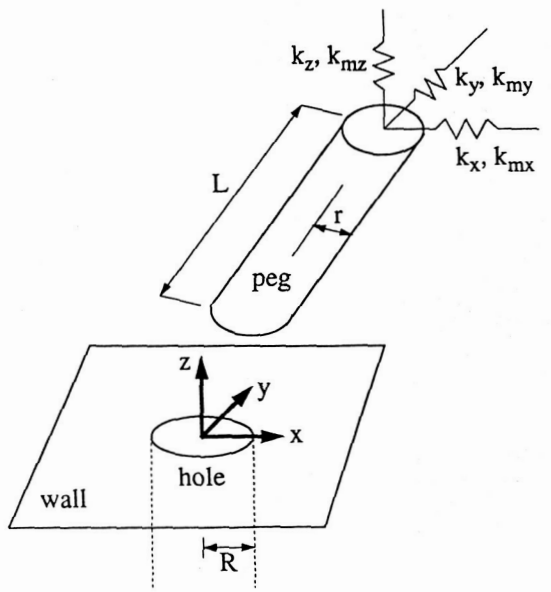


Fig. 5. Geometry of a simulated peg-in-hole task with lateral and angular springs at the compliance center.

IV. PREVIEW DISPLAYS

The success of a telerobotic space operation relies on accurate action planning and verification prior to the actual action execution. This planning/verification capability requirement becomes more significant when dual and/or redundant arm systems are operated in a constrained environment. Preview displays can provide visually perceivable and realistic action planning/verification capability for on-line task execution, thus reducing operation uncertainties and increasing operation safety.

Fig. 6 shows two examples of preview displays to be used for the SMSR operations. The operator first selects an appropriate task segment from the task segment selection menu (lower right panel). When the task segment is selected, the recommended joint angle values for joint 3 and 5 as well as the actual joint angle values are displayed with slider bar displays (upper right panel) for redundancy management. In normal operations, the operator starts with the preview mode before the task execution (upper left panel). There are four options in the preview mode: teleop, auto, record, and playback. In the preview teleop mode, the operator can rehearse the task by teleoperation using graphics simulation without sending the commands to the remote robot site. This teleop rehearsal provides the operator with opportunities not only for practice prior to task execution but also for on-line task planning such as redundancy management, collision avoidance, and sensor planning.

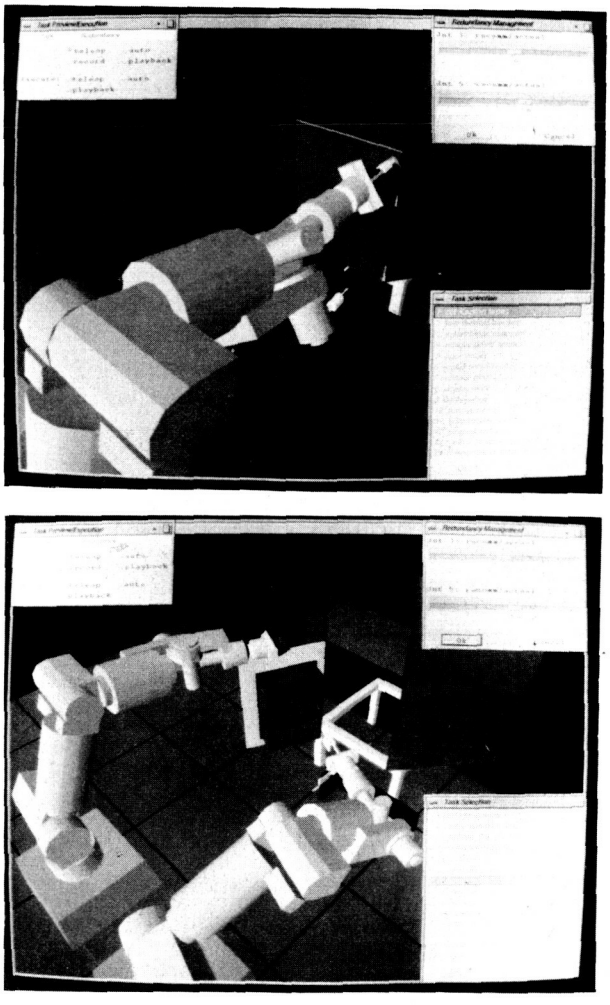


Fig. 6. Preview displays for thermal blanket tape cutting (upper) and electric connector screws removal (lower).

The operator can use the preview auto mode, if an off-line task analysis/planning was done in advance and the pre-planned task sequences are available. In this mode, the operator can see the pre-planned motion as a preview demonstration prior to the actual execution of each task segment. The preview can also record/playback options enable recording and playback of the operator's commands during the teleop rehearsal.

After the preview, the operator selects an option of the task execution mode to actual task execution. In the teleop execution mode, the operator uses manual teleoperation for task execution. In the auto execution mode, the pre-planned motion pre-stored during the off-line task analysis/planning is sent to the remote site for task execution. In the playback execution mode, the recorded motion saved during the teleop rehearsal is sent to the remote site for task execution. If the task segment requires contacts with the environment, manual teleoperation may be preferred due to the lack of accurate calibration of the task environment including cameras. However, if the task segment only needs free-space robot motion without contacts or near collisions, the automatic execution of the pre-planned robot motion may be used for efficiency. An operator intervention capability should be provided during the task execution so that the operator can stop the robot motion at any time when desired.

It is also important to note that a preview graphics display showing a global view of the arm at any desired angle can be very helpful for the operator to visualize the current arm configuration for more comfortable and safer teleoperation in both the preview and the task execution modes. During the task execution mode, the actual joint angles read from the remote robot system are sent to the local site operator control station for graphics update. In our current system, the actual robot joint angles are sent to the graphics system at a data update rate of 30 Hz through a serial I/O line. When the task environment is known, robot arm visualization can be extended to task visualization by showing the operator a graphical simulation of the entire task environment including the robot arm. Other visual cues can also be added, if desired, such as on-the-screen visual enhancements [13] or a graphical representation of a suggested redundancy resolution of a redundant arm.

V. PREDICTIVE DISPLAYS

It is in general difficult for the human operator to control a remote manipulator when the communication time delay exceeds 1 second. The best known strategy to cope with time delay is the "move and wait" strategy [6]. In this

strategy, the operator moves the manipulator a small distance and then waits to see what happens before the next move. When one wants to control a space robot from Earth, there is an unavoidable time delay in the communication link. The round-trip time delay of the communication link between the ground station and a space telerobot in low Earth orbit is expected to be 2 to 8 seconds to relay data via several communication satellites and ground stations. In order to enhance task performance in operating telemanipulators with time delay, we have recently implemented two new schemes at the Advanced Teloperation System: predictive display [2],[9] and shared compliance control [11]. Compliance control is useful during contact or insertion, while predictive display is useful during free-space motion of the the robot arm under quasi-static work environment.

In a predictive display, the graphics model responds instantaneously to the human operator's hand controller commands, while the actual camera view of the arm responds with a communication time delay. Thus the predictive display provides the operator with the non-time-delayed motion of the robot arm graphics model, while the actual robot motion occurs with delay. A predictive display system was originally developed earlier by using a stick figure model of the robot arm overlaid on the actual video image of the arm [15]. Recent advances in graphics technologies enabled us to develop a predictive display with a high-fidelity graphics model which can be either a solid-shaded or a wire-frame model. A high-fidelity graphics model of a Unimation PUMA 560 robot arm was generated and used in our predictive display. A wire-frame model with hidden line removal is sometimes advantageous compared to a solid model. When the wire-frame model of the PUMA arm is overlaid on the camera view, it does not occlude the camera view of the arm.

The real time overlay of the graphics model on the video camera image was achieved by using a video genlock board installed on a Silicon Graphics IRIS-4D/70 GT workstation. The genlock board enables the IRIS graphics output to be synchronized with the incoming video camera signal. It also provides a per-pixel video switching function. Namely, the video output of the genlock board, which is connected to the video monitor for display, can be switched to either the incoming video camera signal or the graphics output signal for each pixel, depending upon the alpha-plane value for the corresponding pixel. In our current application the 8-bit alpha-plane is used simply for graphics image overlay (superimposition), although it allows blending or mixing of two graphics images.

In order to superimpose the PUMA arm graphics model on the camera view of the actual arm, camera calibration is necessary. In our implementation (Fig. 7), camera calibration was achieved by an interactive cooperation between the human operator and the system [9]. The operator provides the correspondences between object model points and camera image points by using a mouse. Thereafter the

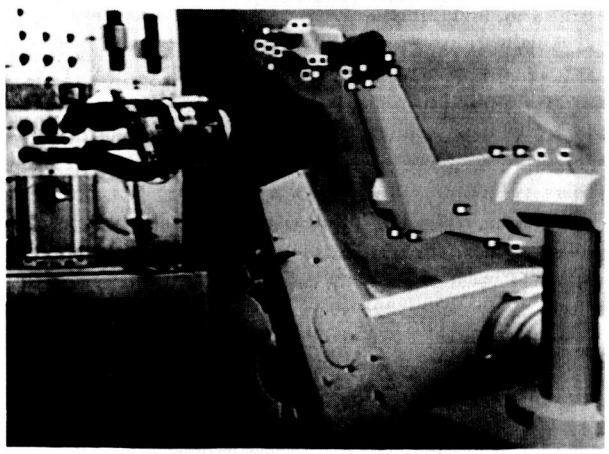


Fig. 7. Visual/manual camera calibration process for graphics overlay. Both the graphics model and the camera view of the PUMA arm appear on the screen. The human operator enters object model points and their corresponding image points using a mouse.

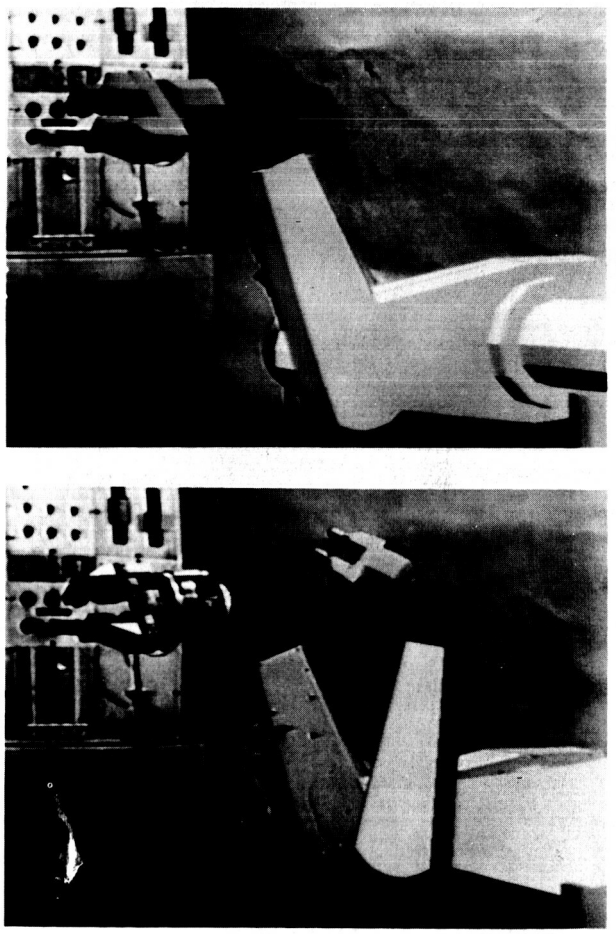


Fig. 8. Calibrated graphics overlay of the solid shaded model on the actual camera view (upper) and a snapshot of a predictive display in operating the remote arm with time delay (lower).

system computes the camera calibration matrix. A linear least-squares method can be used to determine the 12 elements of the 4×3 camera calibration matrix, when 6 or more object points and their corresponding images are given. However, the linear method does not guarantee the orthonormality of the rotation matrix. In our graphics overlay application, the orthonormalized rotation matrix may be preferred. Orthonormalization can be applied after the linear method, but this does not yield the least squares solution. In general, a nonlinear least-squares method has to be employed if we wish to obtain the nonlinear least squares solution that satisfies the orthonormality of the rotation matrix. In the nonlinear method, instead of using 9 elements of a rotation matrix, three angles (pan, tilt, swing) are used to represent the rotation. In our current design, both linear and nonlinear camera calibration algorithms are available. The algorithms above can be used for both cases of when the camera focal length f is known and unknown. The user can select any one of the camera calibration matrix solutions for rendering the PUMA arm graphics model and superimposing on the camera view.

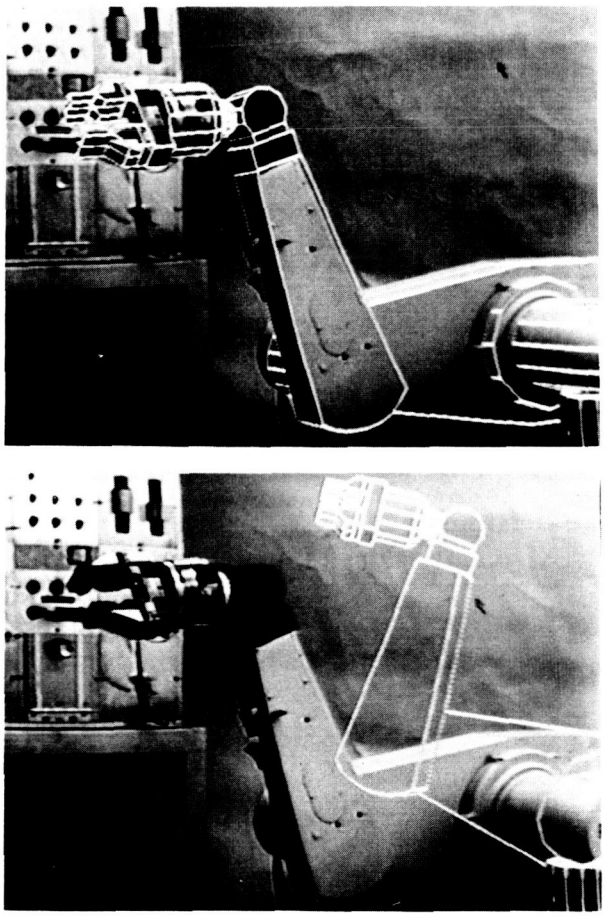


Fig. 9. Calibrated graphics overlay of the wire-frame model on the actual camera view (upper) and a snapshot of a predictive display in operating the remote arm with time delay (lower).

The PUMA arm graphics model superimposed on the actual camera view after the camera calibration is shown in Fig. 8 for the surface model and in Fig. 9 for the wire-frame model. During the actual teleoperation with a predictive display under time delay, the actual camera view of the robot arm follows the graphics model with time delay (Figs. 8b and 9b).

Preliminary experimental results with a simple single-view single-arm tapping task indicate that predictive display enhances the human operator's telemanipulation task performance significantly [2], [8], although it appears that either two-view or stereoscopic predictive displays are necessary for general 3-D tasks.

VI. FUTURE PLANS

Efficiency and reliability of space telerobotic missions will be greatly enhanced through a coherent use of graphics displays in all three phases of off-line task analysis and planning, simulated training, and on-line task execution. New developments and applications of graphics displays for each individual phase described in this paper can be utilized to attain an integrated coherent application of graphics displays throughout all three phases. At present, the simulated SMSR task is being used as a typical exemplar of a space telerobotic mission to demonstrate the effectiveness of our methodology of using graphics displays coherently in all three phases. Experiments will be performed later to quantify teleoperation performance enhancements attained through our methodology. The proposed methodology can be applied to future space telerobotics flight experiments to provide the operator with significant graphics aids during teleoperation.

VII. CONCLUSION

We described new developments and applications of graphics displays in all three phases of off-line task analysis/planning, simulated training, and on-line task execution to reduce operation uncertainties and increase operation efficiency and safety. Task analysis/planning displays provided substantial aid in investigating workcell layout, robot motion planning, and sensor planning for a simulated SMSR task. A force-reflecting training simulator with visual and kinesthetic force virtual reality was developed to serve as an introductory training tool prior to training with actual robots. On-line preview and visualization displays provide the operator with visually perceivable and realistic action planning/verification capability for on-line task execution. Predictive displays provide the operator with a non-delayed graphics model overlaid on the actual camera view to aid the human operator in operating telemanipulators with time delay. These newly developed graphics displays are currently being applied to the simulated SMSR task to demonstrate that an integrated coherent application of graphics displays in all three phases will enhance teleoperation efficiency and reliability.

ACKNOWLEDGMENT

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautic and Space Administration.

REFERENCES

[1] R. H. Adams, A. E. Gross, and C. E. Jennrich, "Remote Repair Demonstration of Solar Maximum Main Electronics Box," Central Research Laboratories, Red Wing, MN, 1987.

[2] A. K. Bejczy, and W. S. Kim, and S. Venema, "The Phantom Robot: Predictive Displays for Teleoperation with Time Delay," IEEE Int. Conf. on Robotics and Automation, pp. 546-551, Cincinnati, OH, May 1990.

[3] A. K. Bejczy, Z. Szakaly, and W. S. Kim, "A Laboratory Breadboard System for Dual-Arm Teleoperation", SOAR '89 workshop, Johnson Space Center: TX, July, 1989.

[4] F. Cepollina and J. R. Jaax, "Extra Vehicular Activity Annex: Solar Maximum Repair Mission", JSC-14082, Annex 11, NASA Johnson Space Center, Mar. 1984.

[5] Deneb Robotics Inc., IGRIP User Manual, Auburn Hills, MI, May 1991.

[6] W. R. Ferrel, "Remote Manipulation with Transmission Delay," IEEE Transactions on Human Factors in Electronics, Vol, HFE-6, Sept. 1965, pp. 24-32.

[7] W. S. Kim, "Developments of New Force Reflecting Control Schemes and an Application to a Teleoperation Training Simulator," IEEE Int. Conf. on Robotics and Automation, Nice, France, May 1992.

[8] W. S. Kim, "Experiments with a Predictive Display and Shared Compliant Control for Time-Delayed Teleoperation," IEEE Conf. on Engineering in Medicine and Biology Society, pp. 1905-1906, Philadelphia, PA, Nov. 1990.

[9] W. S. Kim, "Graphics Overlay and Camera Calibration for Predictive Displays," Jet Propulsion Laboratory Internal Document Engineering Memorandum 347-89-273, Dec. 1989.

[10] W. S. Kim and A. K. Bejczy, "Graphics Displays for Operator Aid in Telemanipulation," IEEE Conf. on Systems, Man, and Cybernetics, Charlottesville, VA, Oct. 1991.

[11] W. S. Kim, B. Hannaford, and A. K. Bejczy, "Force-Reflection and Shared Compliant Control in Operating Telemanipulators with Time Delay," IEEE Trans. on Robotics and Automation, vol. 8, no. 2, 1992.

[12] W. S. Kim and P. Schenker, "A Teloperation Training Simulator with Visual and Kinesthetic Force Virtual Reality," SPIE Conference on Human Vision, Visual Processing, and Digital Display, Society of Photo-Optical Instrumentation Engineering, San Jose, CA, Feb. 1992.

[13] W. S. Kim, F. Tendick, and L. Stark, "Visual Enhancements in Pick-and-Place Tasks: Human Operators Controlling A Simulated Cylindrical Manipulator," IEEE J. of Robotics and Automation, vol. RA-3, no. 5, pp. 418-425, 1987.

[14] S. Lee and A. K. Bejczy, "Redundant Arm Kinematics Control Based on Parameterization," IEEE Int. Conf. on Robotics and Automation, pp. 458-465, Apr. 1991.

[15] M.V. Noyes and T.B. Sheridan, "A Novel Predictor for Telemanipulation Through a Time Delay," Proc. of 20th Annual Conference on Manual Control, NASA Ames Research Center, Moffett Field, CA, 1984.

[16] P. S. Schenker, A. K. Bejczy, W. S. Kim, and S. Lee, "Advanced Man-Machine Interfaces and Control Architecture for Dextrous Teleoperations," IEEE Oceans '91, Honolulu, HI, Oct. 1991.

[17] D. E. Whitney, "Quasi-Static Assembly of Compliantly Supported Rigid Parts," J. of Dynamic Systems, Measurement, and Control, pp. 65-77, 1982.

POTENTIAL LOW-COST TELEROBOTICS FLIGHT EXPERIMENTS

Mr. Brian Wilcox
Jet Propulsion Laboratory
M/S 107-102
4800 Oak Grove Dr.
Pasadena, CA 91109

Few telerobotic devices have flown in space (the remote manipulator system on the Space Shuttle being the principal one that has flown), and proposed telerobotic systems have ballooned in cost to the point where the U.S. has no funded flight telerobotic programs. Three possible alternative systems with limited scope but with significant research and even operational promise are described. These include systems or research in effective control of (possibly flexible) arms in microgravity despite long communication latency, and for inspection of elements of spacecraft such as Space Station Freedom.

JSC FLIGHT EXPERIMENT RECOMMENDATION IN SUPPORT OF SPACE STATION ROBOTIC OPERATIONS

by

Reginald B. Berka, Ph.D.
Head, Robotic Development Section

August 4, 1992

Table of Contents

1 OVERVIEW

1.1 BACKGROUND

The man-tended configuration (MTC) of Space Station Freedom (SSF) provides a unique opportunity to move robotic systems from the laboratory into the mainstream space program. Restricted crew access due to the Shuttle's flight rate, as well as constrained on-orbit stay time, reduces the productivity of a facility dependent on astronauts to perform useful work. A natural tendency toward robotics to perform maintenance and routine tasks will be seen in efforts to increase SSF usefulness. This tendency will provide the foothold for deploying space robots. This paper outlines a flight experiment that will capitalize on the investment in robotic technology made by NASA over the past ten years. The flight experiment described herein provides the technology demonstration necessary for taking advantage of the expected opportunity at MTC.

As a context to this flight experiment, a broader view of the strategy developed at the Johnson Space Center (JSC) is required. In reference to Figure 1, JSC is building toward MTC by developing a ground-based SSF emulation funded jointly by internal funds, NASA/Code R, and NASA/Code M. The purpose of this ground-based Station is to provide a platform whereby technology originally developed at JPL, LaRC, and GSFC can be integrated into a near flight-like condition. For instance, the Automated Robotic Maintenance of Space Station (ARMSS) project integrates flat targets, surface inspection, and other JPL technologies into a Station analogy for evaluation. Also, ARMSS provides the experimental platform for the Capaciflector from GSFC to be evaluated for its usefulness in performing ORU change-out or other tasks where proximity detection is required. The use and enhancement of these ground-based SSF models are planned for use through FY93. The experimental data gathered from tests in these facilities will provide the basis for the technology content of the proposed flight experiment.

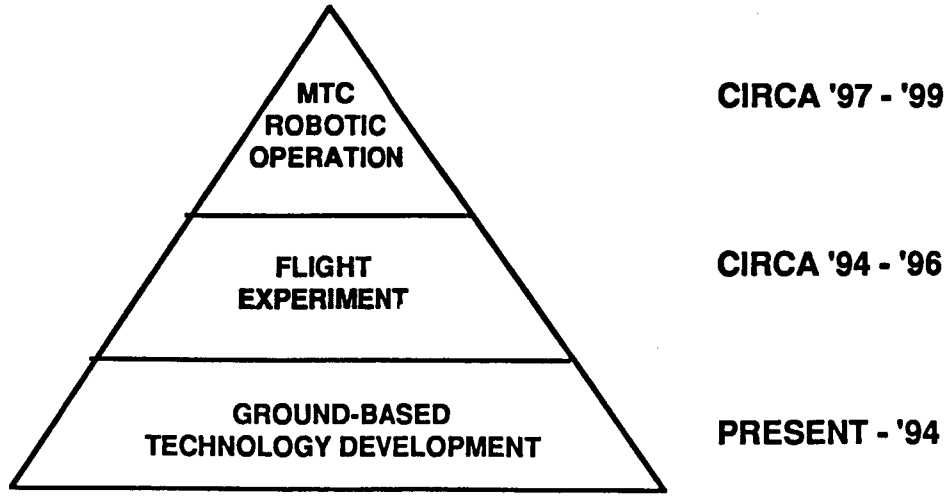


Figure 1 - Technology Progression to MTC

1.2 FLIGHT EXPERIMENT GOALS AND OBJECTIVES

The goal of this proposed flight experiment is to demonstrate the maturity, suitability, usefulness, and availability of robots in performing SSF required tasks. This goal will assist the NASA robotics community in obtaining its broader purpose, to enhance the productivity of Space Station Freedom through the implementation of robotic devices.

To achieve the goals of the flight experiment, several technology objectives must be demonstrated. During the initial man-tended phase of SSF, the ability to perform space operations with robots under remote (ground) control is a necessity. This technology is commonplace in terrestrial applications, but has never been adequately demonstrated in low Earth orbit (LEO). A flight experiment must demonstrate this capability, not in simple robot motions, but rather, in the performance of meaningful SSF derived tasks. Under ground control, the ability to conduct useful space operations needs to be demonstrated in both IVA and EVA environments. This proposal addresses each of these objectives.

1.3 FLIGHT EXPERIMENT DESCRIPTION

The MTC Station can be subdivided into two technology development areas: EVA robotics (EVR) and IVA robotics (IVR). To fully utilize the Station, robotic systems must function in both of these areas under remote control from the ground operations facility. This paper describes an Orbiter flight experiment that addresses these development areas.

Seizing the robotic opportunities of MTC will require SSF robots to be controlled from the ground. This requirement is made difficult by the command and feedback time delays expected in this communication link. Nevertheless, this obstacle must be overcome if robots are to provide the functionality required as full participants in the SSF program. JSC is currently working with the mission controllers to establish a communication path through the nominal mission command links. This path includes the mission control facility, TDRSS, and White Sands. The ground based SSF emulation systems mentioned earlier, namely ARMSS, will be controlled through this communication path. We expect the experiences gained in these ground based experiments to be valuable in addressing the control of robots in low Earth orbit (LEO). Of particular help in solving this problem will be the transfer to JSC of JPL technologies in remote teleoperation and operator coached machine vision. These technologies are planned to be transferred as part of the ARMSS project.

For the proposed flight experiment, JSC proposes the use of its robot ground control system being developed for the ARMSS project. This system is integrated with the mission control network at JSC and can provide the required control capability as well as the proper level of security. Certified crew members will be used to operate the flight experiment from the JSC ground facility. The experiment operations will include various operators performing identical tasks to extract human factors data that is independent of any single crewman. This approach provides a useful baseline of operational data for evaluating experiment performance. LaRC, JPL, and GSFC will be responsible for defining a portion of the experiment, providing a task panel, and training the crew teleoperators in the operation of their respective task panel.

JSC will include the Unversity Space Automation and Robotic Consortium (USARC) in the data analysis process. USARC is a consortium of Texas research universities (Univ. of Texas at Austin, Rice Univ., Texas A&M, and the Univ. of Texas at Arlington) and JSC that are engaged in the development of remote teleoperation technology for use in space. A network has been established that allows an operator at one university to control robots at another. As part of these experiments, UT-Arlington monitors operator inputs and provides human factors analysis. JSC will utilize this capability throughout this experiment to interpret the results from the teleoperation data received during the tests.

Based on JPL remote control technology and requirements from the mission controllers, an advanced ground control station will be designed and constructed at JSC. During the flight experiment, JSC will conduct the mission ground operations through the use of certified crew operators. Each participating Center will be responsible for crew training for their respective segment of the experiment.

1.3.1 EVA ROBOTIC ELEMENT

Currently, JSC and LaRC are involved in the technology capture of the Flight Telerobotic Servicer (FTS) program. The technology capture program provides LaRC with a hydraulic manipulator (that is similar to the flight manipulator), a hand controller, and control software. JSC is managing the continued development of the flight manipulator through final assembly and integration. The goal of this program is to establish a remote communication such that the flight manipulator can be controlled at JSC from LaRC.

The experience gained from this ground experiment with LaRC leads to the recommendation that the FTS be flown in the payload bay of the Orbiter. In this scenario, the FTS is planned to be attached to an MPESS along with a SSF and technology derived task panels developed by the research centers. The ground control and MPESS are critical items in scenario because they both serve to minimize Orbiter integration costs. By performing a SSF task the effectiveness of the technology can be assessed under flight utilization conditions. In constructing this flight element, each Center (JPL, LaRC, and GSFC) would be responsible for a portion of the experiment and the development of their own task panel. These panels would be mounted on the forward end of the MPESS. The Remote Manipulator System would be used to move the task panels to the top of the MPESS, in succession, for manipulation by the FTS.

In addition to the FTS in the payload bay, the flight experiment should include a test of the Autonomous EVA Camera (AERCAM) system. This system provides controllers with a mobile camera that can be optimally placed to support robotic teleoperations. With the myriad of possible external maintenance functions for robots on MTC, this simple "flying camera" provides the support viewing required to perform these tasks under ground control. In addition, surface inspection technology (developed at JPL) can be hosted on this system to examine areas not accessible to baselined robots, such as the solar arrays. Requirements for this system include the ability to be positioned through teleoperation, autonomous station keeping, and autonomous collision avoidance that takes precedence over teleoperator inputs. The AERCAM portion of the flight experiment includes the pre-positioning of SSF surface panels in multiple locations around the payload bay. These panels will depict expected damage resulting from exposure to the

orbital environment. The AERCAM will conduct an inspection of these surfaces through intelligent "wandering" to identify areas of damage. The intelligence embedded into this system will make it a safe and useful partner in conducting on-orbit robotic operations. To minimize the Orbiter integration process and reduce launch costs, this device is configured to fit into the Get-Away-Special (GAS) cannister located on the payload bay longeron. In addition, the AERCAM will be padded to protect the Orbiter from inadvertent collisions.

1.3.1.1 PRINCIPAL INVESTIGATOR

The payload bay element development would be led by JSC. Control software to operate the FTS arm will be delivered from LaRC to JSC as part of the FTS Technology Capture program. Each research center (JPL, LaRC, and GSFC) would be responsible for producing their own task panel based on their own technology requirements and JSC supplied SSF requirements, and for training the crew for the operation of their respective portion of the experiment. JSC would also define interface requirements to the research centers for attachment of the task panel to the MPESS.

1.3.2 IVA ROBOTIC ELEMENT

Many of the scientific and maintenance operations planned for SSF are confined to the habitable volumes of the spacecraft. A robot designed to operate in this environment can provide ground controllers with a productive tool for accomplishing these tasks. JSC proposes to conduct a study of candidate tasks to be performed by an IVA robot. After this survey has been completed, the results will be evaluated to determine if the IVA robot should be included in the flight experiment. If included, this robot would be designed to operate in the pressurized cabin of the Station, or in this case, the Orbiter middeck or SpaceHab. A SSF task panel derived from the survey can be fitted into the foot lockers on the forward wall of the middeck or in the SpaceHab module. These tasks should demonstrate the ability of the robot to perform the series of tasks identified in the task survey.

If, after completing the task study, the IVA robot is included in the flight experiment, special consideration must be made for minimizing Orbiter integration costs. Depending on size and configuration, the robot may be attacted to structural interfaces on the floor of the airlock, similar to the Extravehicular Mobility Unit (EMU, space suit). Alternatively, the robot may be attached directly to the middeck seat interfaces on the floor. The integration details would be finalized as part of the development of this robot.

1.3.2.1 PRINCIPAL INVESTIGATOR

Since the IVA robot would be utilized in the operation of the Station, JSC will conduct a study to determine system requirements.

1.4 FLIGHT EXPERIMENT OPTIONS

Although JSC believes that the flight experiment, composed of EVA, IVA, and ground control components, is the right step for the Agency, fiscal constraints may dictate a more con-

servative approach. With this in mind, the following options are presented in priority order and can be combined in any way consistent with budgetary authority. In all cases, the experiments are proposed to be controlled from the ground, testimony to the importance placed on this technology at JSC.

1.4.1 OPTION 1 - FLIGHT TELEROBOTIC SERVICER

The FTS is selected as the highest priority because of the investment made in its development and the usefulness of its capability. NASA has spent significant resources in the design and development of this manipulator and should follow through with a flight demonstration of its capability. Only through the rigors of the flight experiment will the design decisions made during the development process be validated. This option can include the use of task panels developed at the research centers with each Center providing crew training for their respective equipment. Task panel changeout can be accomplished with the Orbiter's Remote Manipulator System (RMS) demonstrating a cooperative robot task. The proposed cooperative task provides for the FTS to manipulate latches that secure (and release) the task panels after being positioned by the RMS. For example, to remove a task panel, the RMS grapples the panel and waits for the FTS to remove a securing latch before lifting it from the workspace.

1.4.2 OPTION 2 - AERCAM

The AERCAM system is proposed as the next highest priority in this proposal. The needs, identified within the SSF program, for additional camera views to support SSF assembly and long term robotic operations make this project the next highest priority. EVA robotic systems have already enjoyed SSF program support due to success with the Orbiter RMS, the Fisher/Price SSF External Maintenance Report, and international agreements. However, analysis conducted within the program has indicated additional viewing requirements for proposed robotic operations (e.g. assembly, maintenance). The AERCAM system is derived in answer to these needs. The AERCAM can provide bird's eye views of robotic operations, Orbiter approach and berthing, as well as provide inspection capability outside the workspace of existing program robots. Its simplicity and low cost, combined with its ability to enhance existing SSF systems, supports this project as the second highest priority.

For this flight experiment, the AERCAM will be deployed and retrieved using the Orbiter RMS. JSC proposes the use of the Magnetic End Effector (MEE) and Force/Torque Sensor attached to the tip of the RMS to perform these operations. The MEE, developed at JSC, and the Force/Torque Sensor, developed by JPL, are currently set to fly in 1994. Utilizing this system minimizes impacts on the AERCAM design attributed to the grapple and retrieve function.

1.4.3 OPTION 3 - IVA ROBOT

Although the concept of an IVA robot has been discussed in many technical circles, system requirements have not been formalized. However, JSC believes this is an important technology area and a necessity for MTC. A study will be initiated at JSC to better understand the system level requirements for such a device.

1.5 MISSION PROFILE

To conduct this flight experiment, a nominal mission profile for the Shuttle is projected. The MPESS, holding the FTS, would be positioned in the payload bay with the task panels (developed by the participating research centers) on the forward face. The GAS cannister would house the AERCAM system with the IVA robot, if included, attached to a support system in the airlock. The mission profile would not be dedicated to the flight experiment in that other payloads could coexist with the proposed experiment equipment. An Orbiter RMS equipped with the MEE and Force/Torque Sensor, to be used to changeout task panels and stow and deploy the AERCAM, would be required for this flight.

1.6 EXPERIMENT PROFILE

The test of the AERCAM system would occur first in the integrated experiment mission timeline. The AERCAM system would be powered up in the GAS cannister and deployed by the Orbiter RMS utilizing the MEE. While attached to the RMS, a communication and function checkout will be conducted on the AERCAM systems. The AERCAM will be released in a quiescent mode in the payload bay of the Orbiter. In addition to its exterior padding, the energy capacity of the AERCAM will be limited to preclude Orbiter damage if collisions occur. The AERCAM, under ground control, will be flight tested to verify its controllability. During the flight test, the video cameras will be pointed at targets in the Orbiter payload bay and autonomous station keeping tests will be conducted. To further test AERCAM's intelligence, test panels (attached to the payload bay) that depict damage induced by the on-orbit environment will be surveyed and compared against a baseline (no damage) database to test the system's ability to identify damaged SSF areas. At the conclusion of the AERCAM flight test, the system will be returned to a quiescent mode and retrieved, using the MEE, by the Orbiter RMS.

The next experiment phase of the mission will be the teleoperation of the FTS. With a task panel in place, the RMS (with the AERCAM still attached) will position the AERCAM to assist the FTS teleoperation experiments. The FTS experiments on the first task panel will be conducted through ground teleoperation. Following the completion of the task panel experiments, the RMS will release the quiescent AERCAM and changeout the task panel. The RMS will then re-grapple the AERCAM and position it to provide viewing assistance for the FTS experiment. At the completion of the FTS experiment, the RMS stows the AERCAM in the GAS cannister for the return flight.

If included as part of the experiment, the next item in the mission timeline is the IVA robot experiment. For this experiment the crew will configure the robot and task panel for the experiment. Again, through ground teleoperation, the robot will be used to perform tasks on a Station IVA derived task panel. At the conclusion of the experiment, the IVA robot and other test equipment will be stowed on the middeck (or SpaceHab module) for the return flight.

1.7 DATA COLLECTED

The primary purpose of the experiment will be to verify the capabilities of ground teleoperation of space-based robots. Data will be in the form of system operation telemetry as well as video of the experiment. Data will also be taken from the ground control center for analysis of workload, deficiencies, etc. One of the key data sets taken from this flight experiment will be the information regarding task performance by a statistically significant set of crewmembers. This data will depict experiment performance independent of the skill level of a particular crewman. This type of data is necessary before meaningful conclusions can be drawn from the experiment. As mentioned earlier, JSC proposes to utilize the human factors team at the Univ. of Texas-Arlington to assist in the analysis of the experiment data.

1.8 LAUNCH MECHANISM

In preparing for MTC, the Orbiter provides an excellent platform for the development of operational flight systems. Every functional aspect of the MTC Station can be represented in the Orbiter configuration. External robotic activities planned for the Station's transverse boom can be hosted in the Orbiter's payload bay. Robotic systems developed for operations inside the habitable volume of SSF are well suited for the Orbiter middeck. The Orbiter's communication channels through the TDRSS is similar to that expected on SSF, thereby providing the framework for testing ground control. Also, a flight experiment on the Orbiter, where manned spaceflight safety issues must be faced, demonstrates the readiness of robotic technology to be flown on SSF. These factors are the primary reasons for JSC to recommend an Orbiter-based flight experiment.

On this flight, the Orbiter will be required to include the RMS to conduct the EVR aspects of the flight experiment. There are several benefits for utilizing the Space Shuttle for this experiment. First, a complement of Flight Support Equipment (FSE) is available for use. This enables the payloads to be easily integrated in the Orbiter when these standardized systems are used, minimizing integration costs. The use of existing FSE also means that experiment resources ($) can be applied directly to the experiment hardware and not into the development of customized FSE. Also, the crew is available to assist in the experiments. This can be helpful in troubleshooting as well as taking data from the experiment.

1.9 PAYLOAD CHARACTERISTICS

The payload bay experiment equipment will utilize existing Orbiter FSE including a MPESS and a GAS cannister. An IVA robot could be structurally attached to the airlock EMU interface. The FTS experiment, including MPESS and task panels, will weigh approximately

4500 lbs. The AERCAM system will weigh approximately 200 lbs. and be about 28 inches tall with a cylindrical diameter of 19 inches. The IVA robot will also be in the 200 lbs class and located in the Orbiter middeck.

2 TECHNOLOGY DEMONSTRATED

Flight experiments, based on expected program requirements, provide the benefit of focussing individually developed technologies into a single application that is greater than the sum of its parts. However, perhaps the greatest benefit from a robotic flight experiment is the demonstrated solution to otherwise unsolvable programmatic problems. By demonstrating the usefulness, and readiness, of new technology through flight experiments, program managers can include these systems at less program cost and risk.

To control the Orbiter-based robots, JSC proposes the use of advanced teleoperated ground control. The primary technologies included in the ground control of the flight experiment deal with the remote control of robots in a time delay environment through telepresence. Ground based experiments planned at JSC will help define the limitations of remote teleoperation and provide better understanding of the levels of shared control. The shared control technology will be the basis of the ground control telepresence workstation. An important aspect of this ground control system is the integration of this control system with Mission Control at JSC. Mission Control systems utilized to perform this flight experiment provide a precedent for controlling SSF robots at MTC. This section outlines the various technologies and resulting program capabilities for each of the elements of the flight experiment described in this paper.

2.1 EVA ROBOTICS

The FTS arm proposed for the payload bay robotics experiment represents a technology unto itself. It represents a significant investment in technology to provide a flight qualifiable arm for use in space applications. After final assembly and integration of the arm, JSC plans to continue the flight qualification process through onsite environmental testing. In the proposed flight experiment, task panels will be developed by JPL and LaRC to explore such technologies as force reflection in a delayed environment, shared control between the operator and the robot, and control system performance and stabilty. These technologies will be applied to the SSF based task panel to perform surface inspection, ORU changeout (including the GSFC developed Capaciflector), and SSF access door manipulation. The manipulator itself will also be an object for experimental inquiry. The arm, unable to be exercised in a gravity environment, will be investigated for arm kinematic and dynamic characteristics as well as non-linear behaviors in the gear train.

The AERCAM system provides a simple hardware platform for experimentation with sophisticated software algorithms. The hardware is based on a cold gas propulsion system developed at JSC for the Simplified Aid For EVA Rescue (SAFER) project. Added to this existing propulsion system is a complement of stereo and monocular cameras. This component of the flight experiment is required to be teleoperated to a desired position relative to the Orbiter. The teleoperation of this mobile robot will occur under a supervisory software level that avoids collisions with the Orbiter. Once the robotic camera is in the desired position, an autonomous station keeping mode will be engaged. The AERCAM will be required to maintain its position relative to the Orbiter by compensating for orbital

mechanics disturbances. During this phase of the experiment, the station keeping technology must be demonstrated with the Orbiter in Local Vertical, Local Horizontal (LVLH) flight mode, similar to the nominal SSF flight orientation.

2.2 IVA ROBOTICS

The development of an IVA robot could include several advanced technologies. The robot must be capable of teleoperation under autonomous collision avoidance supervision. Also, advanced sensor technologies, used for telepresence feedback and adapted for time delays, are also required in this system to provide input to the control station at JSC.

3 USER INTEREST

The proposed flight experiment produces results meaningful to both near and long term space goals. The robotics research community within NASA can integrate selected technologies into the experiment and verify their usefulness in the space environment. This helps to verify levels of maturity and identify future funding needs and technology priorities. However, perhaps most important is that the technology demonstrated in this flight experiment can enable activities in future lunar and Mars exploration scenarios. The ability to remotely control robots on the moon to perform assembly and maintenance tasks can leverage the available crew time to enhance productivity. In fact, this technology will be an integral component in the success of these future programs.

3.1 COFUNDING/COSPONSORS

Potential sponsors of this flight experiment exist in NASA/Code R as well as other NASA departments. For example, NASA/Code X may have an interest in the enabling robotics technology demonstrated in this experiment. The Space Exploration Initiative may wish to encourage space robotic innovation for use in their proposed lunar and planetary scenarios.

4 PROGRAMMATIC IMPACT

The flight experiment described herein seeks to minimize the STS programmatic impacts. The development strategy with respect to the flight experiment is to design the proposed robots to existing Orbiter interfaces. This enables the systems to be integrated into the launch manifest more easily. Existing FSE is proposed for each flight element as a way of reducing integration costs and focussing resources into the experiment hardware.

5 TELE-ROBOTIC PROGRAM DEVELOPED TECHNOLOGY

The proposed flight experiment is an extension of the current working plan between JSC and other NASA Centers (JPL, LaRC, and GSFC). JSC is developing a high fidelity representation of the SSF system in the laboratory as a means for integrating and evaluating new technologies. The proposed flight experiment would apply those technologies from the research centers that have been integrated into the JSC "ground-based" SSF and integrate them into the flight experiment. This method applies the maximum set of developed technologies that have demonstrated the necessary maturity for flight.

This flight experiment seeks to include relevant technologies developed at the research centers as integral components of the experiment hardware. Some of these technologies have been identified through the technology transfer process currently being accomplished in the ARMSS project. This non-inclusive list of technologies are:

Capaciflector, GSFC

Operator Coached Machine Vision, JPL

Flat Targets, JPL

Remote Site Shared Control, JPL

Surface Inspection, JPL

User Macro Interface, JPL

Magnetic End Effector, JSC

6 POTENTIAL OPERATION IMPACT

The purpose of the proposed flight experiment is to develop technology which can first be used on the Space Station. Its impact would not be on the hardware design of the Station, since the experiment would occur too late in the design cycle. Instead, this experiment would have a profound effect on the operation of the Station. No longer would operation of the MTC Station be confined to the limited periods of crew visits. The flight experiment would provide the Station program with options for using robotic systems to enhance Station operability. The projected experiment time frame aligns favorably with the operations definition cycle that necessarily lags the design cycle.

7 SCHEDULE

7.1 EVA ROBOTIC ELEMENT

7.1.1 FTS MANIPULATOR

Costs for a flight experiment of the FTS Manipulator arm cover a four year effort commencing after the start of FY93. The first years effort (FY93) will (a) define revisions required to the MMAG design of the Data Management Processor System (DMPS) to allow remote controlled operations, (b) provide research of program relevant tasks and sponsors, (c) begin the development of individual task panels by each participating Center to address their interest in the flight experiment, and (d) initiate the development of engineering units for the required supporting electronics and mechanisms. The second year's effort (FY94) will develop the flight components previously prototyped. FY95 will complete the flight system development, integration, testing, and qualification to support a late 1995 flight. After completion of the arm in June of 1993, JSC is committed to an environmental testing program for the Flight Manipulator.

7.1.2 AERCAM

The project will be scoped for technical content during the balance of FY92 and the first quarter of FY93. Design and development, based on successful SAFER testing, will occur during FY93 and into FY94. Assembly, fabrication, testing, and integration of the flight unit would occur during FY94 and part of FY95 with qualification testing, to support the late-1995 flight experiment date.

7.2 IVA ROBOTIC ELEMENT

A task survey of potential uses for an IVA robot will be conducted in the last quarter of FY92 through the first quarter of FY93 at JSC. After completing this review and analyzing the results, TRIWG members will be consulted on the inclusion of the IVA robot in the flight experiment design process.

7.3 GROUND CONTROL

Previous development work at JSC in remote ground control stations would be capitalized on and finalized during the first quarter of FY93. It requires the ability to interface with multiple remote controlled systems, such as the FTS manipulator, the AERCAM, and, possibly, the IVA Robot. Development, build, and test of the ground station would occur during FY93, along with a TDRSS linkage test. A flight support system interface development is required, and would occur during FY94, allowing confirmation of the other flight experiment platforms. Qualification of the flight components would occur in early FY95 to support the late-1995 flight experiment.

7.4 FLIGHT EXPERIMENT SCHEDULE OPTIONS

Each of the elements of the proposed flight experiment are at different levels of readiness. For instance, the FTS Manipulator, originally scheduled as a flight experiment, is nearly 100% designed with fabrication to be completed in mid-1993. The AERCAM system benefits from previous work conducted at JSC in the SAFER program while only definition studies have been initiated on the IVA robot. Therefore, it is possible to move the FTS and AERCAM forward in time, to mid-1995, to meet an earlier flight date. The IVA robot could be flown at a later time. This option is attractive if an early flight date or phased program is desired. The reason for proposing the single flight experiment composed of both EVA and IVA components is reduced cost. By targeting a single flight, the overhead of integrating into multiple launch schedules is avoided.

8 SUMMARY

JSC recommends a Space Shuttle based robotics flight experiment that includes robotic flight elements with ground control by astronauts. The proposal offers configurations for an integrated experiment consisting of an EVA fixed base manipulator, an EVA free flyer, and an IVA robot or subset options thereof. The EVA fixed base manipulator utilizes the flight arm currently being completed in the FTS Technology Capture task and could include coinvestigation by other Centers who would provide experiment subobjectives and associated task panels and who would

also train the astronauts in the performance of their respective portions of the flight experiment. The free flyer element would provide a television camera, would be fail safe, and could be operated in the payload bay under ground control without danger to the crew or the Orbiter. The IVA robotics experiment requires further definition and initial efforts would be used to clarify the value added of such an experiment. Project estimates indicate that a flight date in late 1995 is viable.

Session R7: ROBOTIC MANUFACTURING

Session Chair: Capt. Paul Whalen

N93-32122

RACE PULLS FOR SHARED CONTROL

M. B. Leahy Jr B. K. Cassiday
Robotics and Automation Center of Excellence
Technology and Industrial Support Directorate
Specialized Engineering Branch, Test and Evaluation Section
San Antonio Air Logistics Center, Kelly AFB, TX 78241
ti-race@sadis05.af.mil

Abstract

Maintaining and supporting an aircraft fleet, in a climate of reduced manpower and financial resources, dictates effective utilization of robotics and automation technologies. To help develop a winning robotics and automation program the Air Force Logistics Command created the Robotics and Automation Center of Excellence (RACE). RACE is a command wide focal point. An organic source of expertise to assist the Air Logistic Center (ALC) product directorates in improving process productivity through the judicious insertion of robotics and automation technologies. RACE is a champion for pulling emerging technologies into the aircraft logistic centers. One of those technology pulls is shared control. The small batch sizes, feature uncertainity, and varying work load conspire to make classic industrial robotic solutions impractical. One can view ALC process problems in the context of space robotics without the time delay. The ALCs will benefit greatly from the implementation of a common architecture that supports a range of control actions from fully autonomous to teleoperated. Working with national laboratories and private industry we hope to transistion shared control technology to the depot floor. This paper provides an overview of the RACE internal initiatives and customer support, with particular emphasis on production processes that will benefit from shared control technology.

1 Introduction

In the late 1980s the Air Force commissioned several studies to examine the current and future role of robotics and automation technologies in the Air Logistic Centers (ALC) [8, 1]. Past efforts at technology insertion had been plagued with lack of a cohesive implementation strategy. In addition, total process improvement was often not taken into account. Finally, a lack of in-house expertise forced a complete reliance on contracted efforts. This combination of factors sometimes resulted in technology insertion programs that were overly complex and/or could not perform their intended functions. The Air Force Studies Board report entitled *Advanced Robotics for Air Force Operations* recommended that the Air Force establish a Robotics and Automation Center in a division that would have responsiblity for all robotics related R&D and automation applications [1]. An Acquisition Logistics Division (ALD) report entitled *Robotics Assessment for Logistics Command* recommended that one of the ALCs be assigned as the lead center for robotics technology [8]. The Air Force response to those recommendations was to merge them. In June 1990 the San Antonio ALC (SA-ALC) was given the lead assignment with responsibility for research and implementation of AFLC robotics and technology. The SA-ALC response was to form the Robotics and Automation Center of Excellence (RACE) in February 1991.

The RACE is unique. There is no other organization within the command, or the entire Air Force, which is chartered to perform a similar function. The RACE vision is: *A command-wide focal point. A source of organic expertise ready to assist the product directorates, in improving process productivity by judicious insertion of robotics and automation technology. A champion for the emerging technologies necessary to propel aircraft remanufacturing into the future.* That vision is being brought to fruition under the Technology and Industrial Support Directorate. The core RACE team is in place. A strategic implementation plan (SIP) has been approved and we are hard at work achieving the SIP objectives. A draft Program Action Directorative (PAD) is in the final stages of coordination at Air Force Material Command(AFMC) HQ.

We are supporting customers at each of the five ALCs. RACE team members have also been agres-

sively *selling* the RACE concept inside and outside the Air Force. That sales pitch forms the first part of this paper. In order to perform our mission we must make people aware of our capabilities and objectives. Section two provides a detailed overview of the RACE concept. Section three dicusses our customer support. Current services and future directions are presented. The five major technical areas we are concentrating on are overviewed, with a particular emphasis on the shared control applications. For the purposes of this document we broadly define shared control as the architecture that permits the blending of human and robot actions to accomplish a process. Conclusions are the subject of section four.

2 The RACE Concept

The RACE is the command-wide focal point for robotics and automation technologies. We track all the ALC robotics and automation insertion projects and crossfeed that information to all centers. The six member interdisciplinary RACE team constantly strives to maintain and increase our expertise. Along with understanding the fundamental principals, we are fluent in the current industrial state of the art. We also follow the progress of the academic research community through technical publications and conference attendance. Our objective is to be versed in both the technology and the customer's process. We function as a *robotics solution impedance matcher*, aligning the user requirements with industry capabilities to maximize return on investment. Detailed knowledge of current projects and emerging technologies permits us to specify the most advanced technology suitable to the problem at hand. One of our goals is to pull technology into the depot. Our partners in that technology pull effort are both the national laboratories and industry.

A key tenant of our technology pull effort is to know when to say when. The depots are littered with monuments to technology. The RACE mission is to: *Champion the development and judicious insertion of the robotics and automation technologies necessary to enhance the competitive posture of the ALC product directorates.* We define robotics as the intelligent methodology associated with design, development and judicious insertion of electromechanical manipulators into the industrial base processes. A robot is a reprogrammable electromechanical device with a variety of sensors used to move or manipulate parts and tools in space to achieve a particular task. If a technology component does not improve the quality of the insertion effort, does not contribute directly to the success or evolution of the system, it is inappropriate and must not be acquired. Once a system has been successfully inserted at one depot we work to push that technology into similar processes at other locations. Finally we conduct all of our activities with a look toward the future of the technology and target processes.

The RACE is a customer driven organization. Our primary customers are the product directorate engineers. Those individuals are responsible for enhancing the productivity of a wide variety of industrial processes involved in remanufacturing aircraft. Raw technology doesn't solve their problems. They need sophisticated systems that can be operated by the current workforce. The system must increase productivity and improve the operators working conditions. The shop floor worker must buy into the technology. A successful project is not just meeting some performance specifications, rather the measure of success is whether the project is supported by the workforce and functions as advertised in a production environment. The need to understand our customer and his processes is why the RACE is located at an ALC. Colocation allows us to directly interface with our customers to determine their requirements and act as consultants throughout the life cycle of the insertion project. Our tasking is to provide technical guidance to the product directorates, to ensure their needs are met and competent solutions are produced. We do not perform program management tasks. The product directorates own the processes as well as the associated problems. Ownership of a solution is a critical component of a successful robotics insertion project. We do not make or dictate policy. The customer, who is involved in every phase of the development process, manages the program. When the solution is implemented, it truly belongs to them. Accomplishing that mission will help provide the Air Force with the finest aircraft maintenance and repair facilities in the world.

The validity of the RACE concept is dependent on our ability to improve the acquisition and operation of ALC robotic and automation systems. Our basic premise is to begin solving problems that have a high payback to the government but are also relatively simple to answer. A logical approach of building upon successes is being utilized while the RACE develops the expertise to tackle larger and more difficult problems. With the successful insertion of robotics and automation in those *easy* applications, the culture will shift to one that is more supportive of advanced system development. As the culture shifts we will shift our focus more toward the applied research portion of

our mission.

3 Customer Support

The RACE team provides technical support throughout the life cycle of a robotics project. The level of support depends on the technical background of the product directorate engineer. We can completely offload the technical aspects of the project or simply review a proposal. The usual starting point for our interaction with the process owner is a technical feasibility study. After becoming familiar with their processes, we evaluate alternative solution concepts and assess technical risk and cost. The objective at this stage is not a detailed evaluation, but rather a fundamental analysis upon which to decide if the concept warrents further development of a funding proposal. If the technical and economic aspects of the project are favorable, RACE team members will assist the customer in preparing a proposal to an appropriate funding agency. We will also support that proposal through the review process. Once funding is available our staff can prepare the detailed performance specifications for purchase of off-the-shelf equipment, or the Statement of Work(SOW) and Request for Proposal(RFP), for more complex system procurements. We review contractor bids and provide technical guidance at the required design reviews. For small projects that are pulling an established technology into the ALC, the RACE can eliminate the need for an engineering support contract by using organic engineering resources to perform the implementation.

To assist in the insertion process, Robotics Implementation Working Groups (RIWG), consisting of local project managers and engineers, are being established at each ALC. These groups provide the interface between the RACE and the product directorates. The RIWG members are the eyes and the ears of the RACE to help identify potential robotics or automation applications. The RACE team is marketing a 20 hour seminar series on robotic fundamentals to enhance the background of the RIWG members and other engineers at each ALC. The seminar has been enthusiastically received at SA-ALC and will be given at OC-ALC in September. As current projects progress into the implementation stage we will also assist in the training of the shop floor operators.

3.1 Project Areas

The RACE is currently involved in a variety of projects at each of the ALCs. While the specifics are different, the individual insertion projects can be separated into five major technical thrust areas. A brief overview of large scale systems, industrial automation, and retrofits is presented followed by a more detailed discussion of the two areas that require shared control.

3.1.1 Large Aircraft Systems

The ALCs have a requirement for robotic systems capable of moving painting and stripping tools around the fuselage of all size aircraft. We believe this problem can be subdivided into two catagories based on aircraft size, and a single system developed for each. Those two prototypes should serve as standards around which additional systems are installed as appropriate. There is no technical reason to pay development costs for more than two systems. The Air Force has a working prototype for fighter size aircraft about to enter production at WR-ALC. Other current projects will produce designs that are capable of painting/stripping cargo aircraft. We are working on the SOW for a system capable of painting the C-5. The technology to solve the problem exists. That system will be capable of painting any of the other cargo aircraft in the fleet. The major open technical issue is centered around determining the optimum method for moving a commercial spray painting robot around the surface of the aircraft and the hanger floor.

3.1.2 Industrial Automation

This catagory encompasses a diverse variety of projects where our mission is strictly technology pull. The necessary equipment is commercially available and the basic process has been successfully applied in industry. Our role is to make the directorate engineers aware of the technology and assist in the implementation. Small projects in this class will be completed with organic engineering skills. One example is replacing a manual wench, used to manuver a large heavy x-ray tube, with a large industrial robot. The robot controller will also coordinate the movement of a 2 DOF positioning table. Along with improving the repeatibility and productivity of the x-ray processes, the system will also enhance operator quality of life.

Another example is installing a vision system to measure the clearance between gear teeth and housing on the fuel pump embedded in the fuel control system of a gas turbine engine. The fuel pump is disassembled when the fuel controls are rebuilt. Studies uncovered a clear relationship between the spacing between the gears and housing and the ability of the rebuilt engine to pass inspection. The proposed solution is to

make several key measurements and then properly kit a housing with the apprioriate gear set. The required measurements are to precise to be performed reliably by a human operator.

3.1.3 Retrofits

Several production robotic systems have serious reliability and maintainability problems. We are assisting the product directorates in determining the most prudent course of action in eliminating those problems. Our services range from determining the technical nature of the problem, to proposing a solution, and then assisting in solution implementation. The biggest current effort involves replacement of the Modicon 5200 controllers installed in five paint stripping robots [4] and one automatic derivetor system [2]. The Modicon is no longer in production. Spare parts are very rare and the custom nature of the controller mandated development of all support software from the ground up. These designs were cursed with no evolutionary path for hardware or software upgrades. A complete controller retrofit is required if these systems are to remain in production.

3.1.4 Mobile Robots for NDI

As the age of our cargo aircraft increases so will the requirement for Non-Destructive Inspection(NDI) of large wing and fuselage areas. The inspection process can be repetitious and awkward for an unassisted human. The human is well suited for making judgements about the meaning of sensor readings. A robotic assistant is needed to put the sensor suite into the proper positions. A combination, or sharing, of the process responibilities is required to achieve a near term solution.

The RACE is starting a technical initiative in the area of mobile robots for aircraft skin NDI. Our involvement in this area began in response to a request from SM-ALC for evaluation of an unsolicated proposal for a mobile robot system for aircraft painting. During that evaluation, we discounted the idea for painting. However, the local NDI folks defined a future need for a small mobile system that could carry NDI sensors around the skin of an aircraft. The large robot systems mentioned above are well suited to painting/stripping applications where the process is traditionally accomplished in large dedicated hangers. However, large systems are not appropriate for flightline NDI applications.

The Air Force is not the only government agency interested in airframe NDI. The FAA Aging Aircraft Branch at the FAA Technical Center in Atlantic City, NJ is sponsoring a research project through the Carnegie Mellon Research Institute (CMRI) to develop a prototype mobile robot for eddy current inspection of airframe rivets. The CMRI is the applied research arm of the Carnegie Mellon University (CMU) Robotics Institute.

Following a walk before you attempt to run philosophy, the CMRI team is concentrating on a fairly simple crawler mechanism that will duplicate one of the inspection processes currently performed by USAir NDI technicans. The decision to stick with an existing process and sensing technology provides a robot performance baseline that is sadly missing in many past Air Force projects. Trying to change the process and simultaneously replacing the human operator is usually not a prudent decision. Baselining, with an eye toward future evolution, should enhance operator acceptance and their probability of project success.

The proposed mobile robot is basically a stick with two sets of servo motors and four sets of suction cups. A sketch of the crawler is attached. When performing a scan of a line of rivets, the end suction cups hold the crawler in place and its spine serves as a linear rail for the sensor head to move along. The crawler moves down the rivet line by attaching the sensor slide cups to the surface, releasing the end cups, and then using the linear actuator to move the rail forward. The end cups are attached and the scan process is repeated. The crawler can also move in a vertical direction, but the movement is not as graceful. This device is not designed to move to arbitrary locations on the aircraft surface.

The robotic system will alert inspectors to the presence of abnormal indications and the interpretation will be done by the inspectors. The robot is a tool to assist the operator, not replace him. The robot performs the tedious portions of the task and human provides the judgement skills. The task is divided into components that are ideally suited for the different skills of people and robots. We fully support this operator augmentation (shared) approach to robot system design. More and more people are seeing the virtues of semi-automated systems clearly designed with the human operator in mind. The augmentation approach is the best method to the mobile robot NDI insertion process.

Another major portion of this project is the automated logging of the inspection data. The data acquisition and archiving techniques developed for this project could be applied independent of the robotic system. Access to inspection data would permit ap-

plication of statistical process control to the inspection process.

A prototype is currently being built. Tests conducted with the prototype will determine the feasibility of robotic inspection systems in real world environments. The FAA milestones call for a demonstration of the static scanning processes in November and locomotion tests early next year. If those tests are successful, we will bring the CMRI team down to Kelly to perform some additional tests on the C-5. The long range plan is to circulate videos of the C-5 crawler tests to NDI groups across the command and solicit comments and support for future development. The FAA initiative provides the Air Force with an excellent opportunity to gain a vital new technology with minimal technical risk. The RACE will champion the transfer of this technology from development to ALC specific application.

3.1.5 Telerobotics

Telerobotic technologies will play an increasing part in the solutions to ALC productivity enhancement problems. The small batch sizes, feature uncertainity, and varying workload conspire to make classical industrial robotic solutions impractical for a wide range of depot processes. We need systems that can bridge the gap between manual and complete automation. Shared control technologies provide the material to build that bridge. Telerobotics is a specific subset of shared control. We use the term telerobotics to define a broad class of robotic systems where the actions of the man and machine are tightly coupled. The robotic device responds to human inputs and transfers the human motion into end effector motion. However, unlike teleoperation, the robotic system incorporates some local decision making authority. The basic premise is to augment, not replace the human operator. Blend the individual abilities of each *system*. Humans have superior cognitive and pattern recognition skills, while the robot is a tireless precise positioning system. Thus, the human is still in control but gains the advantages of the machine precision and safety. The prototype center will be expanded to allow investigation of telerobotic concepts. We hope to leverage most of our technology requirements from the developments at the Jet Propulsion Laboratory (JPL). Efforts to form a cooperative working relationship with JPL are ongoing. The latest generation of commercial products also contain the elements upon which a shared control system could be built.

To provide more insight into why telerobotics is a crucial technology in the ALC environment the following several paragraphs highlight a telerobotic solution to an actual depot processes. The Telerobotic Laser Deriveting/Cutting System (TLDCS) is a shared control robotic system used to augment the operator in repairing side and wrap cowlings. Operators will use the TLDCS to derivet a damaged area on the cowling, cut out the damaged section, and then cut out an identically sized replacement patch. Technicians in the Aircraft Directorate repair over 450 side and wrap cowlings each year. Cowling repair times range from 60-450 hours depending on complexity. Roughly one third of the repair time is spent on deriveting and the cutting and triming of new aircraft skins. Removing one rivet is not a difficult job, however repeating that task a hundred times a day is physically demanding, tedious, and time consuming. Removing damaged skin sections is also a demanding labor intensive process and existing sheet metal cutting machines are not designed to quickly reproduce unique patterns. Adding to the task difficulty is the large size of some parts, which often forces the repair technician into awkward and potentially dangerous positions. Other hazards include the high noise environment and the razor sharp chips produced during drilling and cutting. The objective of this project is to increase the productivity and improve the worker quality of life for the aircraft skin repair process.

Projects to improve the deriveting process have been previously attempted. The Navy initiated a mobile derivetor project in 1983 [5]. The project objectives were too ambitious for the existing technology base and the end product was a large, complex system with limited flexibility that was not well received by shop personnel. The REPTECH office funded the development of a Robotic Deriveting and Drilling Cell installed at OO-ALC [2]. This system was designed to remove all of the rivets from a rigidily fixtured, stiff, flat workpiece. While the system is considered a success, the hard automation and mechanical drilling makes it unsuitable for quick removal of random rivet patterns from engine side cowls, a major workload at SA-ALC. A more serious limitation is the robot control system. The controller is no longer in production and duplicating this deriveting system would require an extensive software rewrite. Neither of those systems could be easily modified to include the ability to trim or cut sheet metal.

After studying the previously developed systems, it became evident that an alternative to conventional drilling was required. The solution to the problems of mechanical drilling of rivets (rapid tool wear, high part forces, spinning rivets) is laser drilling. Laser technol-

ogy is routinely used to drill holes in sheet metal products. Once enough material is removed from the center of the rivet it is a simple operation for the human operator to punch out the rivet. For small workloads the expense and added complexity required to automatically punch out the rivet is not justified. Laser drilling is ideally suited for robotic application due to the non-contact nature of the task and the light payload applied to the end-effector from the flexible fiber optic cable used to channel the beam. Laser drilling also eliminates the noise and razor share chip hazards. The operator will be removed from the immediate worksite and the laser completely enclosed in a separate structure for safety.

An additional key benefit of the laser approach is the ability to use the laser to perform cutting operations. The laser system will be an industrial Nd:YAG (Neodymium:Yitrium, Aluminum, Garnet) solid state laser. Working with the laser vendors and in-house laboratory technicans we will conduct a series of experiments to characterize the heat effects of the laser and optimize our spot size and power accordingly. A flexible fiber optic cable will channel the laser beam to the end-effector of the robot. Cameras and proximity sensors will also be attached to the end-effector. Additional cameras will be mounted at appropriate locations inside the laser booth. The laser will be completely enclosed for operator safety. Safety interlocks will be in place to preclude laser firing while the operator is positioning the workpiece inside the booth.

The TLDCS will have the capability, similar to a conventional water jet cutter, to cut a unique pattern without operator programming or CAD data. The operator will only have to highlight the pattern on the surface and then the TLDCS will visually follow the pattern and cut out the part. The ability to augment the operator during the skin cutting and triming process is crucial to the economic success of this project.

A fully automated system is inappropriate for the varied size and placement of the workpieces. The TLDCS will be designed using the principles of shared control. System capabilities will range from a completely teleoperated mode, where the human operator precisely positions the laser over each rivet, to a semi-automated mode where he/she simply identifies the target rivets on a graphics screen and then the robot slaves to those positions and awaits the command to shoot. Individual operators will be able to select the mode that allows them to be most productive. Expensive fixturing is eliminated. Multipass teaching methods are not required. The operator simply puts the workpiece within the robot workspace and proceeds to derivet or cut. No more advance preparation is required than picking up a hand drill or mini grinder.

Deriveting and cutting are mandatory operations in the repair and replacement of all aircraft skins. While this project is specifically directed toward repair of side and wrap cowls, the TLDCS has the flexibility to remove rivets from any aircraft component and the cutting operation could be extended beyond sheet metal repair. The only limitation would be the physical size of the part. Large size parts would require a larger robotic system, but the basic architecture of the TLCDS would remain unchanged. This is **not** a one-of system, but rather a true prototype. The process of laser deriveting and cutting is just the first instantiation of generic telerobotic workstation. By placing the design emphasis on the controller architecture and operator interface we create a system that forms a baseline for all future depot telerobotic applications. The TLDCS architecture is suitable for additional applications ranging from spray painting and stripping to fuel tank sealant application. The successful completion of this project will pave the way for low cost judicious insertion of telerobotics technology throughout the depots. Telerobotics is a critical robotics and automation technology for improving the quality of depot processes.

3.2 Future Directions

The RACE is pursuing several additional initiatives to improve our customer support capability. A brief overview is provided below. For more details consult [7].

The development of a prototype center is a critical element. That center will contain several state-of-the-art commercial robots equipped with vision and force sensor systems. The robots will be used for: prototyping, education, and detailed system design. Funding for those systems has been secured and the procurement process is underway. Funding to procure a advanced robot system capable of evaluating technology emerging from the laboratory has been requested. Another key element of the prototype center is a graphical simulation system. That requirement has been identified, and is at the top of our unfunded requirements list.

We are also championing several technologies necessary to reduce robotic and automation system insertion expenses. Our objective is to foster the development of robotic systems that fit the PC analogy; an inexpensive, flexible platform which has a large support and software base and is easily customized for a particular application. To accomplish that vision, the

ideas of reconfigurable software, modular robots, and standardized interfaces must be more fully developed.

4 Conclusions

The Air Force Material Command Robotics and Automation Center of Excellence (RACE) is open for business at SA-ALC. The RACE provides a command-wide focal point for process engineers who seek to improve efficiency through the judicious insertion of robotics and automation technology. Major initiatives are underway to transform the RACE concept into reality. RACE is a champion for pulling emerging technologies into the Aircraft Logistics Centers. Shared control is a key tenant in those technology pull activities. The RACE team is currently providing technical support to process engineers at all five ALCs. Projects range from painting the C-5 to automating a simple X-ray system. The RACE team is ready to work with you. Join us as we champion competitive processes through intelligent machines.

References

[1] Air Force Studies Board, *Advanced Robotics for Air Force Applications*, National Academy Press, Washington, D. C., 1989.

[2] G. S. Bettencourt, Robotic Deriveting and Drilling Cell, *Proceedings of REPTECH 89*, Oklahoma City, OK, August 22-24, 1989.

[3] H. Berry, Robotic Aircraft Painting with SAFARI, *Proceedings of the SME Conference on Maintaining and Supporting an Aircraft Fleet*, Dallas, TX, June 9-11, 1992.

[4] E. A. Franke, Automated Paint Stripping, *Proceedings of REPTECH 89*, Oklahoma City, OK, August 22-24, 1989.

[5] E. A. Franke, Update on the Robotic Derivetor, *Proceedings of Fastec West '86*, Anaheim, CA, Oct 21-23, 1986.

[6] S. Hofacker, Large Aircraft Robotic Paint Stripping, *Proceedings of the SME Conference on Maintaining and Supporting an Aircraft Fleet*, Dallas, TX, June 9-11, 1992.

[7] M. B. Leahy, Jr. and B. K. Cassiday, Towards RACE, *Proceedings of the SME Conference on Maintaining and Supporting an Aircraft Fleet*, Dallas, TX, June 9-11, 1992.

[8] *Robotics Assessment for Logistics Command*, ALD Report, ALD/LTT, WPAFB, OH, Jan 1990.

AUTOMATED ASSEMBLY CENTER (AAC)

Lt. Robert J. Stauffer
Manufacturing Technology Directorate
Wright Laboratory (AFMC)
Wright-Patterson AFB, OH 45433-6533

PROBLEM: The aerospace assembly environment is characterized by low production rates, frequency product definition changes, and highly complex assembly structures, all compounded by paper-based data systems. Recent integration initiatives—such as the Automated Airframe Assembly Program (AAAP), the Flexible Assembly Subsystems (FASs), and the ongoing CALS program—have demonstrated exceptional approaches for generating, transferring, and applying product definition data. It is now apparent that many of the pieces necessary to establish an integrated automated assembly environment exist or are emerging. However, the establishment of these technologies in the industrial base or the production of systems has been slow, resulting in lower productivity and quality and wasted defense dollars.

SOLUTION: The development of an Automated Assembly Center (AAC) program is a key initiative to tie together the technologies developed under past assembly-related programs. This integration of past successful programs will provide a focus for ongoing work. Government action and funding are necessary to lead the establishment of a product and process information-driven defense manufacturing base. The solution is an implementation program to facilitate the transfer of integrated process and product information technologies.

OBJECTIVE: The objective of this project is to integrate advanced assembly and assembly support technology under a comprehensive architecture, implement automated assembly technologies in the production of high-visibility DOD weapon systems, and document the improved cost, quality, and lead time. This will enhance the production of DOD weapon systems by utilizing the latest commercially available technologies combined into a flexible system that will be able to readily incorporate new technologies as they emerge. Automated assembly encompasses product data, process planning, information management polices and framework, three schema architecture, open systems communications, intelligent robots, flexible multi-ability end effectors, knowledge-based/expert systems, intelligent workstations, intelligent sensor systems, and PDES/PDDI data standards.

AN OVERVIEW OF THE KENNEDY SPACE CENTER ROBOTICS PROGRAM

SIXTH ANNUAL
SPACE OPERATIONS, APPLICATIONS, AND RESEARCH SYMPOSIUM
AUGUST 6, 1992
L.B. JOHNSON SPACE CENTER

Eric L. Rhodes
Advanced Technology Office
Robotics Manager

KSC is Applications oriented.

Primary mission – operational use

Offer the capability to prove robotics work
on the ground
before being used in space.

KSC ROBOTICS & AUTOMATION PROJECTS

- ✔ TILE ROBOT (Tessellator)
- ✔ HFIR (HEPA Filter Inspection Robot)
- ✔ ARID (Automatic Radiator Inspection Project)

- SENSOR BASED OBSTACLE AVOIDANCE FOR REDUNDANT MANIPULATORS ("SKIN")
- SELF-CONTAINED DEPLOYABLE SERPENTINE TRUSS
- KINESTATIC PLATFORM
- CG DETERMINATION

- PAYLOAD INSPECTION ROBOT STUDY

TILE ROBOT (Tessellator)

A semi- autonomous robotic system to rewaterproof and inspect thermal protection systems tiles on the underside of the orbiter.

DESIGN FEATURES: 11 degrees of freedom

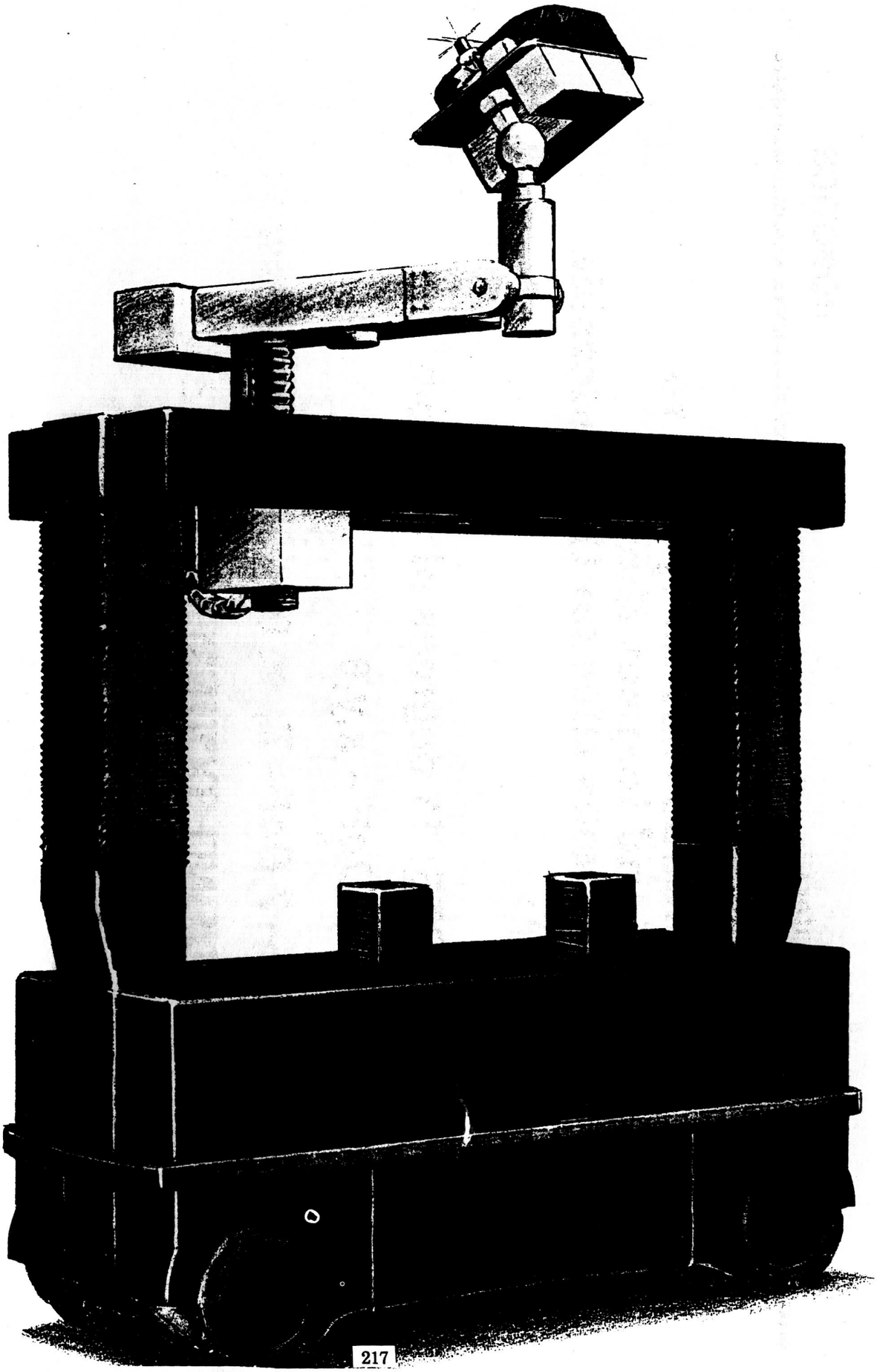
MOBILE BASE - 3 DoF – x,y,θ

MANIPULATOR – 7 DoF– Z; z, θ; x; pitch, roll, yaw

REWATERPROOFING SYSTEM – 1 DoF – Z

VISION SYSTEM

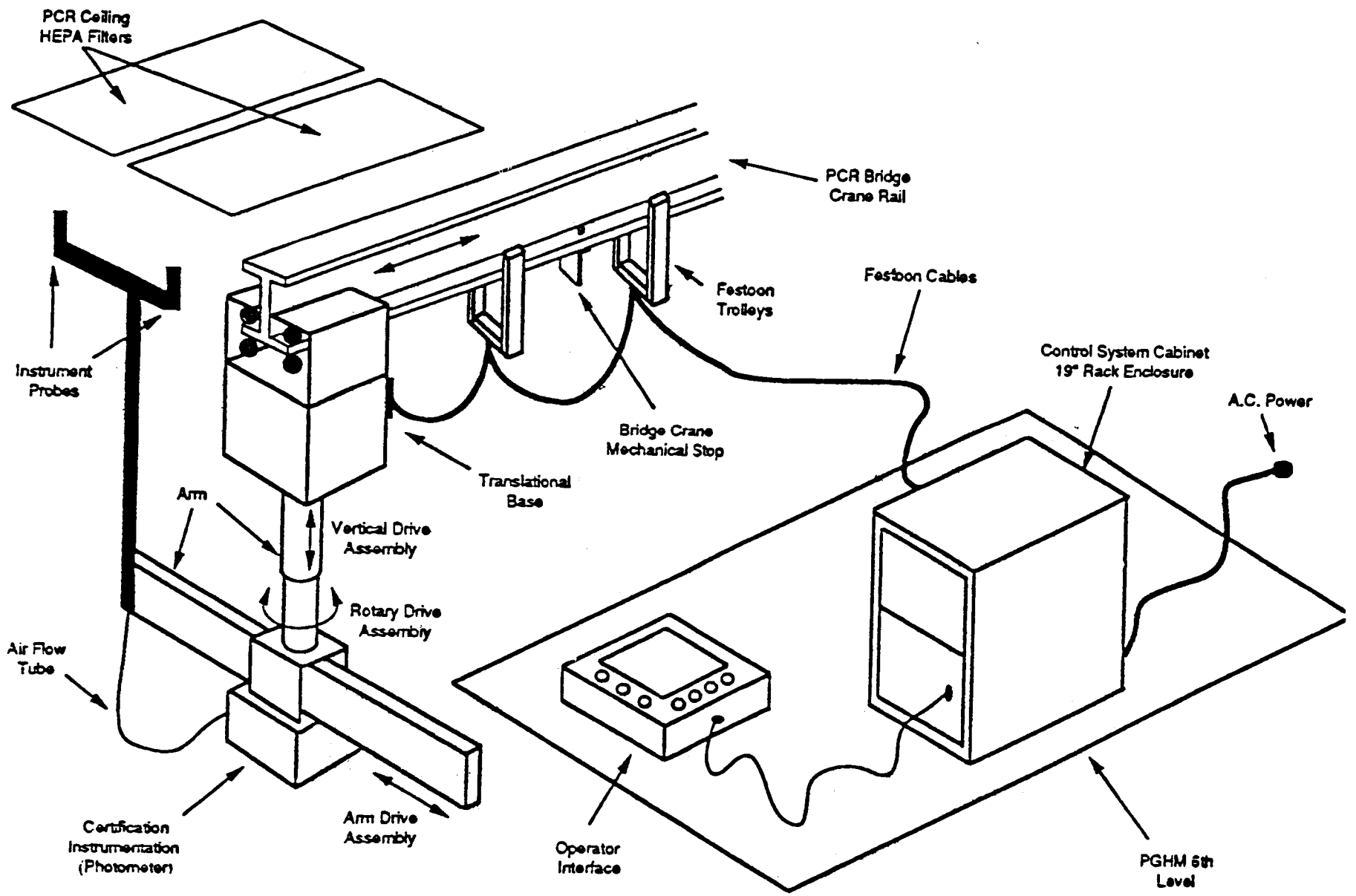
WORKCELL CONTROLLER



AUTONOMOUS INSPECTION OF HEPA FILTERS LOCATED AT THE TOP OF THE LC 39 PAYLOAD CHANGEOUT ROOMS

DESIGN FEATURES

- 4 DEGREES OF FREEDOM – x, y,z,θ 3 PRISMATIC, 1 ROT
- USES EXISTING BRIDGE CRANE RAIL
- PORTABLE FOR USE AT PADS A & B
- END EFFECTORS
  - PARTICULATE COUNTER
  - AIR VELOCITY PROBE
  - LASER DISPLACEMENT SENSOR
  - CCD COLOR CAMERA


PCR Ceiling
HEPA Filters
PCR Bridge
Crane Rail
Festoon
Trolleys
Festoon Cables
Control System Cabinet
19" Rack Enclosure
A.C. Power
Instrument
Probes
Bridge Crane
Mechanical Stop
Translational
Base
Arm
Vertical Drive
Assembly
Rotary Drive
Assembly
Air Flow
Tube
Certification
Instrumentation
(Photometer)
Arm Drive
Assembly
Operator
Interface
PGHM 6th
Level

ARID (Automatic Radiator Inspection Project)

AUTONOMOUS INSPECTION OF ORBITER RADIATORS FOR DAMAGE WHILE IN THE ORBITER PROCESSING FACILITY

ARID DESIGN FEATURES

- RADIATOR DAMAGE DETECTED BY DIGITIZED IMAGE COMPARISON
- FRAME SHIFTING TO ACCOMODATE ORBITER / DOOR POSITIONING TOLERANCES
- IMAGE SIZE 4" X 4"
- INSPECTION TOOL SPEED 2.6 IN / SEC
- ESTIMATED INSPECTION TIME 4.5 HOURS

NO STEP
ITIC
TIONS
ENT LAB

The KSC Advantage

KSC OFFERS A LOCATION FOR ROBOTS
TO PERFORM WORK ON GENUINE SPACE HARDWARE.

WHERE

- ROBOTIC SYSTEMS CAN BE PERFECTED
- MISTAKES CAN BE CORRECTED
- ROBUSTNESS CAN BE PROVEN

ON THE GROUND IN A SAFE, SHIRT SLEEVE ENVIRONMENT.

ON THE DESIGN OF FAULT-TOLERANT ROBOTIC MANIPULATOR SYSTEMS

Dr. Delbert Tesar
University of Texas at Austin
Mechanical Engineering Department
Austin, TX 78712

Robotic systems are finding increasing use in space applications. Many of these devices are going to be operational onboard the Space Station Freedom. Fault tolerance has been deemed necessary because of the criticality of the tasks and also due to the inaccessibility of the systems to ready maintenance and repair.

Design for fault tolerance for manipulator systems in an area within Robotics that is without precedence in the literature. In this paper, we will attempt to lay down the foundations for such a technology.

Design for fault tolerance demands new and special approaches to design, often at considerable variance from established design practices. These design aspects, together with reliability evaluation and modeling tools, are presented. Mechanical architectures that employ protective redundancies at many levels and have a modular architecture are then studied in detail. Once a mechanical architecture for fault tolerance has been derived, the chronological stages of operational fault tolerance are investigated. Failure detection, isolation, and estimation methods are surveyed, and such method for robot sensors and actuators are derived. Failure recovery methods are also presented for each of the protective layers of redundancy. Failure recovery tactics often span all of the layers of a control hierarchy. Thus, a unified framework for decision-making and control, which orchestrates both the nominal redundancy management tasks and the failure management tasks, has been derived. The well-developed field of fault-tolerant computers is studied next, and some design principles relevant to the design of fault-tolerant robot controllers are abstracted. Conclusions are drawn, and a roadmap for the design of fault-tolerant manipulator systems is laid out with recommendations for a 10 DOF arm with dual actuators at each joint.

SECTION II

AUTOMATION AND INTELLIGENT SYSTEMS

Session A1: ADVANCED KNOWLEDGE-BASED SYSTEMS TECHNOLOGY

Session Chair: Dr. Abe Waksman

AIR FORCE KNOWLEDGE-BASED SYSTEMS BASIC RESEARCH PROGRAM

Dr. Abe Waksman
AFOSR/NM
Bolling AFB, Building 410
Washington, D.C. 20332-6448

Abstract unavailable at time of publication.

ADVANCED ARTIFICIAL INTELLIGENCE TECHNOLOGY TESTBED

Mr. Craig S. Anken

Air Force Material Command
Rome Laboratory
Griffiss AFB, NY USA
13441-5700

Abstract

The Advanced Artificial Intelligence Technology Testbed (AAITT) is a laboratory testbed for the design, analysis, integration, evaluation and excercising of large-scale, complex, software systems, composed of both knowledge-based and conventional components. The AAITT assists its users in configuring various problem-solving application suites; observing and measuring the behavior of these applications and the interactions between their constituent modules; gathering and analyzing statistics about the occurrence of key events; and flexibly and quickly altering the interaction of modules within the applications for further study.

1.0 Introduction

The importance of "intelligent", large-scale military decision support systems for effective command and control is becoming more and more apparent. In time, decision aids will be prevalent throughout every aspect of military operations, aiding in decisions that will have major impacts on the battle. These systems will be composed of both knowledge-based and conventional modules, and will interact as part of some predefined problem-solving strategy. The ability to iteratively define this strategy, integrate, and deploy these suites thus becomes very important.

An intelligent command and control (C^2) decision aid often begins its life in a "sterile" laboratory environment. Each decision aid is developed to solve a relatively narrow problem within the overall C^2 decision-making/ management problem. To provide the user guidance, this aid has sources of data and knowledge stored in local format(s), a problem solving methodology which uses the knowledge to reason over the data, and usually the ability to interact with the user. Testing may be performed by presenting the aid with predefined scenarios and comparing the results to some standard. While this approach may be adequate in the laboratory setting, what happens when this tool is brought into the operational realm? That is, when it is brought into an environment where information may be stored in several databases and where multiple decision aids and/or conventional programs are working at various portions of the overall problem. Finally, how reliable will this system be, having been developed in isolation thousands of miles from the battlefield? To answer some of these questions, the Rome Laboratory has embarked on the development of the AAITT, a distributed environment that will support the integration, testing of, and cooperation among multiple decision aids.

2.0 Advanced AI Technology Testbed (AAITT)

Before describing the AAITT, several terms need to be defined. A component is a stand-alone software program designed to solve part of an overall problem (see Figure 1). User-supplied components (USCs) refer to those components which the application developer supplies and wants connected to the testbed. A module is a component with some type of software "wrapper" that allows it to communicate with other modules. An AAITT application is a collection of modules configured to work interactively to solve an overall problem.

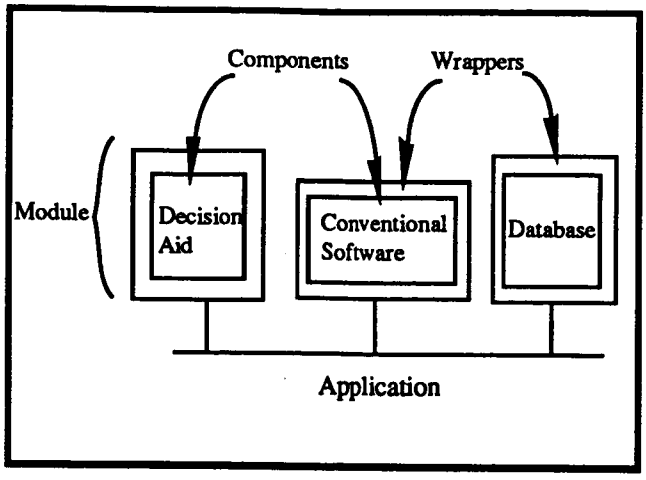


Figure 1. Testbed terminology

The goal of the Rome Laboratory Advanced AI Technology Testbed is to develop a decision aid/conventional software integration environment which will allow the user to:

1. Easily configure various application suites by providing tools to add or remove user-supplied components, or to modify the communication paths of the various problem solving modules,

2. Provide generic database and simulation modules that can be tailored to the needs of the current application,

3. Observe and trace these modules' actions and interactions,

4. Select, gather, and review metrics and statistics on run-time performance, and

5. Rapidly change the flavor of the interactions among the suite's components based upon the results of previous runs.

The components of the AAITT, shown in Figure 2, consist of the testbed manager (later described as the MCM Workstation), a database, a simulator, and various USCs.

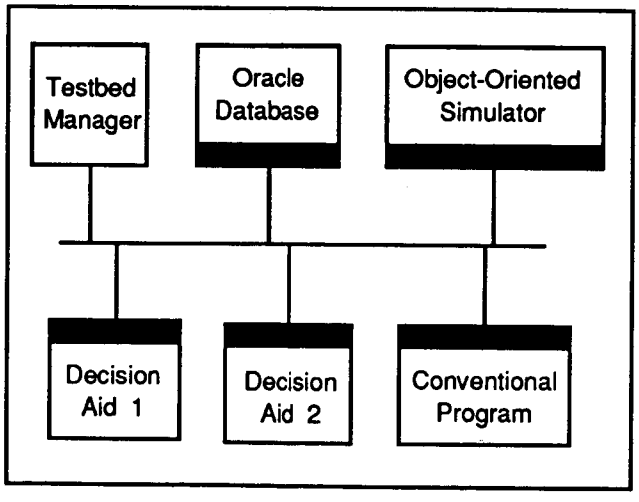


Figure 2. Network View of AAITT Application

The grayed sections in the figure above represent interface software. A software controlled architecture (soft architecture) has been chosen for the AAITT. A soft architecture is one that can be graphically configured and modified by a testbed manager, while minimizing the recoding of the interface software by hand. An example

configuration is shown in Figure 3. To try a different communication architecture (e.g., a blackboard approach instead of data-flow), the user would make the necessary changes using the graphical interface provided by the testbed manager.

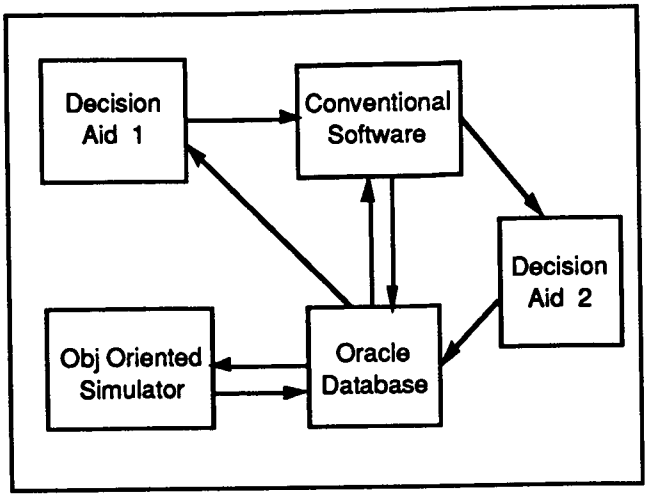


Figure 3. Experimental AAITT Architecture

To support the soft architecture envisioned by the AAITT, three major sub-systems are being developed. These are the Distributed Processing Substrate (DPS), the Module Framework, and the Modeling, Control, and Monitoring (MCM) Workstation.

2.1 DPS

The Distributed Processing Substrate will allow dissimilar software systems (e.g., databases, simulations, expert-systems, and conventional software) running on heterogeneous hardware platforms (e.g., VAX, SUN, and Symbolics) to interact with one another. An important feature of the DPS is that is will translate data representations between the various hardware systems and programming languages. Several candidates were considered by the contractor team for providing the foundation of the testbed and are shown in Table 1.

Product	Developer
Alpha	Carnegie Mellon Univ.
Cronus	BBN Corp.
ISIS/Meta	Cornell University
Mach	Carnegie Mellon Univ.
MetaCourier	Symbiotics, Inc.
Star O/S	ADS Corp.

Table 1. Candidate Environments [GE-91]

The criteria for selection was:

1. Runs on Sun 3/4 and Symbolics 36xx.

2. Runs on Vaxs running VMS and ULTRIX.

3. Runs with applications written in C/C++ and Common Lisp.

4. Runs with applications written in Ada.

5. Provides interface specification and stub code generators.

6. Provides application building tools.

7. Stable, mature product.

The resulting evaluation of each candidate is shown in Table 2.

Product	Alpha	Cronus	Isis/ Meta	Mach	Meta Courier	Star O/S
Sun3/4 Symbolics	No	Yes	No	No	Yes	Yes
Vax,VMS Ultrix	No	Yes	No	No	Unknown	No
C/C++ Lisp	No	Yes	No	No	Yes	Yes
Ada Interface	No	Future	No	Yes	No	No
Int. Spec & Stub Code Gen.	No	Yes	No	Yes	Yes	No
Appl Bld. Tools	None	Excl.	Good	Good	Good	None
Stability	Poor	Excl.	Excl.	Excl.	Fair	Excl.

Table 2. Candidate Env. Evaluation [GE-91]

Based upon this analysis, the Cronus distributed computing environment [BBN-89] was selected. Cronus provides heterogeneous host support for distributed application development. It gives the user an object-oriented view of resources on a network. In addition to this object-oriented view provided by Cronus, the DPS will use ABE [Erman-90], providing a module-oriented programming capability. This will allow the structuring of a distributed testbed application at a higher level than Cronus. The ABE environment supports modules which are independent entities that communicate data and control among themselves through very well-defined interfaces. The goal is to use the higher-level, computational and architectural model provided by ABE with the lower-level, distributed system environment support provided by Cronus.

2.2 Module Framework

The Module Framework will allow the user to easily embed a new component in the AAITT. This framework is used to create Component Interface Managers (CIMs). A CIM is a wrapper that provides the interface between the distributed testbed and each component. The Module Framework will facilitate the creation of the CIM by guiding the user through the creation process and by providing a library of generic CIMs. It will also provide a semi-automated generator for easily creating and modifying CIMs. CIMs will also allow the testbed to control module loading, initialization, resetting, etc.

There are several types of communication that will occur at a given CIM. These types include simple reception and transmission of data, and may include more complicated transmissions which then wait for a response. When transmitting, the CIM must know which module to send the message to. The name of the destination module will be set when the user completes the modeling phase at the MCM Workstation. The recipient can easily be changed by making the appropriate changes at the workstation. (Note: the actual location on the network of the receiving module will be maintained, transparently to the user, by the DPS.) CIM to CIM message types include ASCII text, integer, real, and database query response.

2.3 MCM Workstation

The Modeling, Control, and Monitoring Workstation will provide a central user console for building and configuring applications, and for the display and analysis of performance metrics gathered during application runs.

The modeling functions will allow a user to specify graphically how the modules are to interact with one another during execution. This will define the architecture of the application. The control functions will allow the user to load, initialize, execute, and reset the components distributed over the testbed's network. Break point capabilities will aid in the debugging process. The monitoring functions will allow the user to specify measurements, monitors, and instrumentation. Measurements refer to the quantifiable features that help the user understand system performance. Monitors are the procedures through which measurements are captured. Instrumentation allows the user to process the resulting measurements and present them in an appropriate manner. The monitoring capabilities of the MCM Workstation will be very important in testing out the USCs and set the AAITT apart from testbeds that simply try to connect existing systems.

2.4 Testing

As previously stated, it is very important to understand the strengths and weaknesses of the USCs before providing them to operational units. These weaknesses may include actual errors, problem size limitations, excessive run times, incompatibilities, and so on. The AAITT will have features to address the problem of verification and validation of USC's. The questions, is the software doing the job right, and is the software doing the right job, will need to be answered.

"Job Right"

Determining whether the SW is doing the job right will provide the end users with some confidence that the USCs will work as advertised. To verify the results of the USCs, the AAITT will provide metrics for evaluation and heuristics. Timing and memory limitations will be addressed by these metrics. In addition, bottlenecks will be identified, possibly indicating that a new application architecture should be selected or that faster computers/networks should be used. Higher level metrics will also be addressed, for instance the number of mission routes planned per minute, or the speed at which the simulation simulates one hour of time.

The quality of a solution cannot be determined by the speed of the inference engine or the number of queries handled by the database per minute. Quality can sometimes be assessed by using some type of heuristic. To support this, the AAITT will need some type of generic heuristic evaluation module that can be tailored to evaluate a given class of USCs. For instance, a route planner heuristic evaluation module may take the planned routes and determine for each route if the specified aircraft can fly at the chosen altitudes, if the aircraft has enough range, given the available on-board fuel, and whether the legs of the route are too short (difficult to follow) or too long (vulnerable to enemy attack). This evaluation module could be used regardless of which specific route planner was being tested.

"Right Job"

In determining whether the software is doing the right job, end users need to make sure the USCs really address their particular problems. To validate whether or not the USCs are meeting their needs, the testbed will provide rapid prototyping facilities and the benefit of graphical simulations.

The testbed will allow component developers the ability to quickly demonstrate their systems in a realistic environment. By providing both database and simulation capabilities, developers can concentrate on software algorithms and user interface issues, instead of developing sets of problems and ways to evaluate results. This will mean that USC users can be brought in earlier in the development process, allowing them to direct or redirect the developers before design decisions are made. This process will help developers better understand the users' needs, not just the users' stated requirements.

The simulation component will provide a simulation language and various tools for developing graphical simulations. This will allow users to monitor the execution of plans developed by the USCs. The simulation language is designed to be interactive so that user developed simulations can be stopped, queried, modified, and continued at any point. This type of interactive simulation should help in determining whether the USC developer has really addressed the user's needs.

2.5 AAITT Support Components

The AAITT will have generic components to facilitate the development and testing of USCs. These generic components have been used to create demonstration components to plan missions in the tactical air combat domain.

Conventional and "expert" software work by understanding the problem at hand, automatically or semi-automatically attempting to solve the problem, and then posting the results (usually to a screen or file). Databases and simulations can provide a supporting environment for evaluating these types of software systems. A database can be used to present the USCs with specific problem scenarios. Once the USCs finish processing, the results of the components can be stored in the database for later evaluation. Heuristics can be helpful in evaluating the quality of a result, for instance the routes chosen by a route planner could be evaluated on the percentage of time that allied aircraft are exposed to enemy air defenses. If a heuristic is not available, a simulation could be helpful, allowing a human evaluator to see the results of following the recommendations of a particular USC. Generic components will be provided to develop these types of databases and simulations.

The testbed will have a generic database for storing and retrieving information needed by the USCs. Results of the USCs can be stored in the database for later evaluation. The Oracle relational database has been chosen as the testbed's database shell. Using the DPS, the database will appear as a module in the testbed that can respond to structured query language (SQL) statements. As additional databases are created, they can then be used by other USCs.

The testbed will have a generic simulation capability to show the consequences of following the results of the other modules. The ERIC object-oriented simulation language has been chosen as the testbed's simulation language [Hilton-90]. ERIC is based on the Common Lisp Object System (CLOS) and allows the simulation developer to develop a hierarchy of object classes with associated behaviors and attributes. ERIC allows objects to be created dynamically and for the simulation to be halted, modified, and continued without recompiling. The message parser provided by ERIC allows for expressive message passing. Results from the simulation can be posted to the database or sent on to USCs.

2.6 Demonstration Components

Demonstration components will be used during the AAITT demonstrations to show the feasibility of the testbed.

TAC-DB

The Tactical Database (TAC-DB) provides a realistic, though unclassified, laydown of tactical units and equipment in the central European theater [Kearney-90]. This database was developed by Knowledge Systems Corporation (KSC) in 1989 and reflects the NATO and Warsaw Pact capabilities at that time. In addition to identifying where units are located, the database contains information on individual weapon systems. As new USCs are added to the testbed with new requirements for domain information, TAC-DB will be extended to provide the necessary support.

TAC-DB provides the "blue-view" of the world for the USCs. This means that information in the database is only as accurate as blue intelligence is. This makes sense since USCs should not be able to access more information than the blue side knows. (Note: there may be some "red-view" information stored in the database used only by the simulator to create red units, etc., that are currently unknown to the blue side.)

LACE

The Land Air Combat in ERIC (LACE) simulation simulates the tactical engagement of NATO and Warsaw Pact forces in the central European theater [Anken-89]. The simulation reads in friendly and enemy unit information from TAC-DB, creates these objects, and then executes air tasking orders (ATOs) also stored in the database. Friendly air missions must penetrate through enemy air defenses in order to attack targets. As missions return, their results are posted to TAC-DB for use by the USCs.

LACE contains the ground truth for the scenario. The USCs do not have direct access to this ground truth data, only to the TAC-DB. The USCs could, however, be used to schedule LACE reconnaissance missions which will post their intelligence reports to TAC-DB upon completion. This information then becomes available to the other modules of the testbed.

AMPS / RPDW

Two decision aids will be included as part of early demonstrations. These include the Air Force Mission Planning System (AMPS) developed by The MITRE Corporation and portions of the Route Planner Development Workstation (RPDW) developed by the Jet Propulsion Laboratory. AMPS [Dawson-90] was developed to support the planning and replanning of ATOs. ATOs are the plans used by NATO to conduct air missions. AMPS contains domain knowledge concerning aircraft and weapon capabilities, weapon system availability, and scheduling constraints. In addition to planning these missions, AMPS was developed to intelligently support replanning, the process of fixing a plan which has problems, without replanning the whole ATO.

The RPDW [Cameron-85] was designed to facilitate the development and evaluation of route planning algorithms. Our route planner is one of the algorithms provided by the RPDW. To use this planner, the user must first develop a threat-contour map based on the location and line of sight of the various air defense systems. The planner then determines a route through the map while attempting to minimize aircraft vulnerability. The results generated by AMPS and the route planner will be sent to TAC-DB and then on to the simulation. The idea is to provide feedback to the decision aid developers concerning their systems.

3.0 Summary

C2 decision aids and conventional programs are being developed to support operational commanders. An environment is needed to allow these dissimilar systems to easily work together. In addition, realistic testing of these components is required to make sure they perform as expected. The AAITT is designed to address these concerns by providing a loosely coupled, distributed support environment, allowing independently developed aids to cooperate. The generic database and simulation capabilities will allow for easy development and evaluation of C2 decision support systems. If the database is large enough and the simulation can provide updates at fast enough rates, the testbed can present the decision aids with realistic wartime environments. This is the kind of interoperability and evaluation that is needed before fielding composite operational systems.

REFERENCES

[Anken-89] Anken, C.S., "LACE: Land Air Combat in ERIC," RADC-TR-89-219, October 1989.

[BBN-89] BBN Systems and Technologies Corp., "Cronus Advanced Development Model," RADC-TR-89-151 Vol. 1, September 1989.

[Cameron-85] Cameron, J., Cooper, B., Mishkin, A., "Route Planner Development Workstation, Volume 2 - Software User's Guide," JPL D-2733 Vol 2, U.S. Army Engineer Topographic Laboratories, Jet Propulsion Laboratories, November 1985.

[Dawson-90] Dawson, B., Day, D., Mulvehill, A., 90 "The AMPS," RADC-TR-90-131, July 1990.

[Erman-90] Erman, L., Davidson, J., Lark, J., Hayes-Roth, F., "ABE: Final Technical Report," TTR-ISE-90-104, May 1990.

[GE-91] GE/ATL, "Advanced AI Technology Testbed, Technical Information Report (DPS) Analysis Draft," F30602-90-C-0079, February 1991.

[Hilton-90] Hilton, M. and Grimshaw, J. "ERIC Manual," RADC-TR-90-84, April 1990.

[Kearney-90] Kearney, H., Lazzara, A., Pendergast, J., Richer, A., Erich, R., "Cooperative Red and Blue Database System," RADC-TR-90-98, June 1990.

KNOWLEDGE-BASED SIMULATION USING OBJECT-ORIENTED PROGRAMMING

Karen M. Sidoran

Air Force Materiel Command
Rome Laboratory
525 Brooks Rd.
Griffiss AFB, NY
13441-5700

ABSTRACT

Simulations have become a powerful mechanism for understanding and modeling complex phenomena. Their results have had substantial impact on a broad range of decisions in the military, government, and industry. Because of this, new techniques are continually being explored and developed to make them even more useful, understandable, extendable, and efficient. One such area of research is the application of the knowledge-based methods of artificial intelligence (AI) to the computer simulation field.

The goal of knowledge-based simulation is to facilitate building simulations of greatly increased power and comprehensibility by making use of deeper knowledge about the behavior of the simulated world. One technique for representing and manipulating knowledge that has been enhanced by the AI community is object-oriented programming. Using this technique, the entities of a discrete-event simulation can be viewed as objects in an object-oriented formulation. Knowledge can be factual (i.e., attributes of an entity) or behavioral (i.e., how the entity is to behave in certain circumstances).

Rome Laboratory's Advanced Simulation Environment (RASE) has been developed as a research vehicle to provide an enhanced simulation development environment for building more intelligent, interactive, flexible, and realistic simulations. This capability will support current and future battle management research and provide a test of the object-oriented paradigm for use in large scale military applications.

INTRODUCTION

The adoption and incorporation of the various knowledge-based methods of artificial intelligence (AI) into the simulation process provides benefits in a number of areas. Some of these areas include: knowledge representation in the simulation model, decision making within a simulation, rapid prototyping of models, data analysis of simulator-generated outputs, and model modification and maintenance (Nielsen, 1991).

The problem of representation is central to all of AI. Eight paradigms that have found favor in today's practice are [Garcia-91]:

- Semantic networks
- Frames and scripts
- Procedural representations
- Analogical or direct representations
- Specialized languages for knowledge representation
- Object-oriented programming
- Logic representations
- Rule-based representations

This paper will focus mainly on the technique for representing and manipulating knowledge known as object-oriented programming. In many practical situations, the eight representation paradigms previously mentioned are combined in hybrid systems. An example of this integration will be discussed in a latter section of this paper which discusses an extension of our object-oriented research to incorporate a rule-based approach.

OBJECT-ORIENTED PROGRAMMING

The object-oriented approach attempts to manage the complexity that is characteristic of real-world problems by abstracting out information and encapsulating it within objects [Wirfs-Brock-90]. Using this technique, the entities of a discrete-event simulation can be considered as objects in an object-oriented formulation. There are two types of knowledge associated with each object, factual and behavioral. An object's factual data can be viewed as the attributes of the entity, characterizing its capabilities, current state, and parameters. The object's behavioral knowledge characterizes how the entity is to behave in certain circumstances. Messages sent to an object can provide factual data (such as recalling an attribute) or initiate a specific behavior (such as asking an air mission to fly its route).

This type of representation and processing supports rapid prototyping in that particular model functions can be changed or replaced with minimal impact on other sections of the model. It also facilitates the modification and maintenance of models. Often we can organize objects into a family of homogeneous classes, which are mutually distinct, but do share certain fundamental properties in common. Within the object-oriented paradigm, a class definition can specify from which classes a new family of objects will inherit (i.e., automatically obtain factual and behavioral knowledge). This saves having to copy or rewrite common variables and methods and helps maintain consistency when code modifications and model extensions are made [Zeigler-90].

ROME LABORATORY'S ADVANCED SIMULATION ENVIRONMENT (RASE)

Since 1986, Rome Laboratory (RL) has been exploring ways to utilize the advantages offered by the object-oriented paradigm to improve Air Force battle management simulation [Anken-91]. The focus of RL research has centered on building an environment that will allow us to develop and test object-oriented simulations.

The Rome Laboratory Advanced Simulation Environment (RASE) began with the development of an object-oriented language known as Enhanced ROSS [McArthur-82] in Common Lisp (ERIC)[Hilton-90]. To exploit the capabilities of ERIC, the program was expanded to include the development of an object-oriented Cartographic system (CARTO), a Map Display System which serves as the graphical interface to the CARTO system, and an interactive object-oriented battlefield simulation known as the Land Air Combat in ERIC (LACE) simulation. Other support tools were also developed and include an object hierarchy browser (HIER), an object editor and run time instance viewer (RACK), and a simulation clock manipulation tool (Clock Viewer) [Anken-91]. Currently, RASE is geared toward the tactical battlefield domain and includes a cartographic system which covers a portion of Central Europe. However, many of the RASE tools are generic and can be used to develop simulations in other domains.

LACE, which was built as a part of, and with the tools of the RASE environment, simulates the tactical engagement of NATO and Warsaw Pact forces in Central Europe. The simulation's primary focus is on the air side of the battlefield model. While the main purpose of LACE has been to evaluate the object-oriented paradigm in a large-scale military application, it has also been designed to provide a realistic environment for evaluating tactical decision aids and to support tactical commanders in their decision making process.

LACE currently includes objects for:

- air-facilities
- wings & squadrons
- fighter, cargo, & refueling aircraft
- surface-to-air missile (SAM) sites
- runways
- petroleum-oil-lubricant facilities (POL)
- munition targets

Missions include offensive counter-air, transport, refueling, and SAM Suppression. The overall system includes hundreds of stationary and mobile objects able to interact in the simulation. The current demonstration includes six air missions with 32 SAM units and radars set up to defend enemy targets.

With continual map display updates the simulation runs approximately 38 times faster than real-time. Running LACE without graphics has resulted in speed-ups of over 70 times real-time [Anken-91].

CURRENT RESEARCH ISSUES

Through our work with RASE we have found that object-oriented programming lends itself very well to battlefield simulation. The modern-day battlefield is a complex domain with thousands of autonomous agents. Each agent, or object, has its own goals and behaviors while functioning within the goals and plans of higher entities. Object-oriented programming provides a natural means of representing these entities.

We feel that our research efforts have effectively addressed many of the problems with traditional military simulations. Most of these simulation environments lack features essential to adequately support Air Force needs. They lack flexibility, are difficult to embed knowledge into, are expensive to build and maintain, and generally are not compatible with current knowledge-based decision aids [Anken-89]. In contrast, the RASE environment provides a highly interactive, intelligent, and flexible simulation capability.

Through our development of RASE we have identified other areas of potential research for improving simulation techniques. One such area is improving the inspectability of our simulation models. Without the capability to thoroughly inspect and understand the underlying model, it is difficult to validate (ensure that the right system was built) and verify (ensure that the system was built right) the model that is being or has been developed.

Another potential area of improvement is execution speedup through distributed processing. If simulations could be made to run more rapidly, users could experiment with more battlefield strategies in a shorter length of time. These issues will be briefly discussed in the following sections.

COMBINING RULE AND OBJECT-ORIENTED APPROACHES

While the object-oriented approach provides inspectability in terms of the attributes of the entities being modeled and how entities are related within the model, there is no information explicitly available in terms of the model behavior (i.e., the when, where, why, and how's of the events associated with object actions). Thus, the logic of why an event occurs, or does not occur, is lost within the body of program code. Inspecting simulation models, such as LACE, usually requires the user to review the source code to fully understand the interactions between simulated entities. Understanding the results of the simulation is often accomplished by saving (usually to disk or hard copy) selected data items as the simulation progresses.

This lack of explicit knowledge severely limits the types of questions traditional state-based simulations can answer. These limitations are most evident in the area of model inspectability where the user would require the simulation to explain its behavior in meaningful terms.

Understanding the results of the simulation model includes statistical and causal information [Grimshaw-92]. Statistical information includes both the history of the simulation and derived historical information. History would include both events and temporal state information. For example, in the LACE model, the user may want to know what happened to a given air mission on its way to the target or the location of that air mission at a certain time. Derived historical information might include averages or standard deviations of events and temporal states. The user may want to inquire about the average expected target damage or the chances of a mission being successful.

Causality refers to the events or states which lead to the occurrence of a specific event. One may want to ask why a certain air mission was shot down or why another mission was successful. Using the object-oriented approach of RASE, this information was not readily available. Queries were limited to "What if" questions, allowing the user to interactively stop the simulation, make changes to object attributes, create or destroy objects, and restart the simulation to see what happened.

To overcome these limitations, the combined use of a rule-based approach with an object-oriented approach was proposed [Grimshaw-92]. In this hybrid approach, time-lines are used to capture temporal values and causal events, providing history and causality to the simulation. The modeling language that was developed

at Rome Laboratory to demonstrate this concept is an extension to ERIC known as DeclERIC (Declarative ERIC). The work in this area incorporates the advantages of both rule-based and object-oriented techniques, providing the ease of representation with the explicit representation of behavioral knowledge.

Time-lines are created that capture what an object's selected attribute values will be for the length of the simulation assuming there are no interactions between objects. For example, if a SAM site has 5 missiles initially, we assume it will have 5 missiles forever. If an aircraft is given a route, we assume it will fly this route without any change in course or speed. Then, leveraging off TMM (Time Map Manager, a system developed by Tom Dean) [Dean-85; Dean-87], the programmer defines Spatial-Temporal Possibilities (STP's) of interest [Grimshaw-92], such as when a mission will be over its target, or when a mission is going to be in-range of a SAM site.

Using the STP's and the time-lines as pre-conditions, DeclERIC will create all the possible bound rules from the user-defined rule base. A rule example is the SAM-fire-rule which states that if a mission Y is in-range of SAM X, it's jammer is off, and SAM X has more than one missile, then decrement the number of missiles of SAM X and update the status of mission Y if it has been damaged.

After binding all the rules it can, DeclERIC picks the earliest one, discarding the rest, and executes it, changing whatever object attributes that need to be changed according to the postconditions of the executed rule. Then it cycles through, sending TMM the new time-lines, receiving a new set of STP's, and finding the next rule to fire. DeclERIC is finished when there are no more rules that can fire. After the simulation has executed, the user can watch it play on a graphical display and ask menu-driven questions about what happened and why. The value of object attributes now reflect their full history and a list of rules that have been fired is maintained by DeclERIC. Work in the area of declarative simulation and DeclERIC is still underway.

DISTRIBUTING AN OBJECT-ORIENTED SIMULATION

A second issue that was previously mentioned is that of simulation execution speed. Although the object-oriented paradigm made LACE easier to change and analyze than previous simulations, it was discovered that execution time could still present a problem when thousands of objects must interact in a very large simulation. This opened another avenue of research which has been to study the feasibility of extending an object-oriented simulation into a distributed paradigm. During this project, DERIC, a distributed version of ERIC, was developed at Rome Laboratory [Lawton-92].

The development of DERIC has shown that this extension into the distributed/parallel paradigm is feasible [Lawton-91]. As a result of this research, several properties of ERIC were identified that must be preserved in a distributed extension, such as causality and user interactivity. Also, a number of critical issues inherent to parallel simulations, including deadlock avoidance, atomicity, and event monotonicity were encountered [Lawton-92]. While it has been concluded that distributed computing extensions to ERIC are desirable, speedups of distributed object-oriented simulation languages are not easily realized [Lawton-91]. Research in the area of distributed object-oriented simulation is still actively in progress.

CONCLUSION

Knowledge-based simulation has the potential to provide a powerful mechanism for understanding and modelling complex phenomena. The goal of this area of research is to build simulations that are more powerful and more comprehensible by making use of deeper knowledge about the behavior of the simulated world. Such knowledge is usually omitted from traditional simulations since they are unable to utilize it. This lack of explicit knowledge severely limits simulations and makes it difficult to verify the correctness of their model, makes them difficult to comprehend, and restricts the kinds of questions they can answer. A general lack of formal validation techniques has limited the use of knowledge-based simulation in critical applications. This has contributed to the fact that the number of operational systems is significantly less than the number of systems developed. However, the need for "intelligent" simulations is evident and knowledge-based paradigms such as object-oriented programming and rule-based inferencing provide the mechanism for advancing the development of computer simulation models in many domains.

REFERENCES

[Anken-89] Anken, Craig S., "LACE: Land Air Combat in ERIC," Rome Lab In-House report RADC-TR-89-219, Rome Laboratory, Griffiss AFB, New York, October 1989.

[Anken-91] Anken, Craig S., "An Air/Land Combat Simulation and Scenario Generation Assistant," PROCEEDINGS OF THE 1991 INTERNATIONAL SIMULATION TECHNOLOGY CONFERENCE, Orlando, Florida, October 21-23, 1991.

[Dean-85] Dean, T., "Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving", Tech. Report 433, Computer Science Department, Yale University, New Haven, Connecticut, 1985.

[Dean-87] Dean, Thomas L. and McDermott, Drew V., "Temporal Data Base Management", AI Journal, Elsevler Science Publishers, 1987.

[Garcia-91] Garcia, Oscar N., and Chien, Yi-Tzuu, KNOWLEDGE-BASED SYSTEMS: FUNDAMENTALS AND TOOLS, IEEE Computer Society Press, Los Alamitos, California, 1991.

[Grimshaw-92] Grimshaw, Capt Jeffrey, "Simulation Modeling Using an Integrated Rule and Object-Oriented Approach," Unpublished Rome Lab In-house Report, Rome Laboratory, Griffiss AFB, New York, January 1992.

[Hilton-90] Hilton, M., and Grimshaw, J., "ERIC Manual," Rome Lab In-House Report RADC-TR-90-84, Rome Laboratory, Griffiss AFB, New York, April 1990.

[Lawton-91] Lawton, James H., Krumvieda, Clifford D., "Distributing Object-Oriented Simulation Languages," PROCEEDINGS OF THE 1991 SUMMER COMPUTER SIMULATION CONFERENCE, Baltimore, Maryland, July 22-24, 1991.

[Lawton-92] Lawton, James H., Krumvieda, Clifford D., "DERIC: A Distributed Object-Oriented Simulation Language," NATO WORKSHOP ON OBJECT-ORIENTED MODELING OF DISTRIBUTED SYSTEMS, DREV, Quebec, Canada, 1992.

[McArthur-82] McArthur, David and Klahr, Philip, "The ROSS Language Manual," N-1854-AF, The Rand Corporation, Santa Monica, 1982.

[Nielson-91] Nielson, Norman R., "Application of Artificial Intelligence Techniques to Simulation," KNOWLEDGE-BASED SIMULATION METHODOLOGY AND APPLICATION, Vol. 4, Springer-Verlag New York Inc., New York, New York, 1991.

[Wirfs-Brock-90] Wirfs-Brock, Rebecca, et. al, DESIGNING OBJECT-ORIENTED SOFTWARE, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[Zeigler-90] Zeigler, Bernard P., OBJECT-ORIENTED SIMULATION WITH HIERARCHICAL, MODULAR MODELS, Academic Press Inc., San Diego, California, 1990.

GENERATION AND EXPLORATION OF AGGREGATION ABSTRACTIONS FOR SCHEDULING AND RESOURCE ALLOCATION

Dr. Michael R. Lowry[1] and Dr. Theodore A. Linden[2]
[1]AI Research Branch
M.S. 269-2
NASA/Ames Research Center
Moffett Field, CA 94035-1000
[2]Advanced Decision Systems
A Division of Booz, Allen & Hamilton
Mountain View, CA 94043

This paper presents research on the abstraction of computational theories for scheduling and resource allocation. The paper describes both theory and methods for the automated generation of aggregation abstractions and approximations in which detailed resource allocation constraints are replaced by constraints between aggregate demand and capacity. The interaction of aggregation abstraction generation with the more thoroughly investigated abstractions of weakening operator preconditions is briefly discussed.

The purpose of generating abstract theories for aggregated demand and resources includes: answering queries about aggregate properties, such as gross feasibility; reducing computational costs by using the solution of aggregate problems to guide the solution of detailed problems; facilitating reformulating theories to approximate problems for which there are efficient problem-solving methods; and reducing computational costs of scheduling by providing more opportunities for variable and value-ordering heuristics to be effective. Experiments are being developed to characterize the properties of aggregations that make them cost effective.

Both abstract and concrete theories are represented in a variant of first-order predicate calculus, parameterized multi-sorted logic that facilitates specification of large problems. A particular problem is conceptually represented as a set of ground sentences that is consistent with a quantified theory.

Session A2: PLANNING AND SCHEDULING I

Session Chair: Lt. Jennifer Skidmore

PROTOCOLS FOR DISTRIBUTIVE SCHEDULING†

Stephen F. Richards

Associate Professor, Ambassador College

Computer Information Systems Department

Big Sandy, Texas 75755

Barry Fox

Planning and Scheduling Technology Group

McDonnell Douglas Space Systems Company - Houston Division

16055 Space Center Blvd.

Houston Tx 77062

ABSTRACT

The increasing complexity of space operations and the inclusion of interorganizational and international groups in the planning and control of space missions lead to requirements for greater communication, coordination, and cooperation among mission schedulers. These schedulers must jointly allocate scarce shared resources among the various operational and mission oriented activities while adhering to all constraints. This scheduling environment is complicated by such factors as the presence of varying perspectives and conflicting objectives among the schedulers, the need for different schedulers to work in parallel, and limited communication among schedulers. Smooth interaction among schedulers requires the use of protocols that govern such issues as resource sharing, authority to update the schedule, and communication of updates. This paper addresses the development and characteristics of such protocols and their use in a distributed scheduling environment that incorporates computer-aided scheduling tools. An example problem is drawn from the domain of space shuttle mission planning.

†Supported in part by the NASA/ASEE Summer Faculty Fellowship Program, Johnson Space Center 1992

INTRODUCTION

Scheduling is the process of assigning resources and times to each activity of a plan (or plans) while ensuring that each constraint is obeyed. Optimization criteria can determine the relative desirability of two alternate schedules. Although scheduling problems are often simple to visualize and express, scheduling is an NP-complete problem, so attempts to apply mathematical programming to scheduling have met with very limited success [2, 6]. In fact, programming approaches have been limited to very narrow problem domains, especially that of the job-shop, in which jobs must be assigned to various machines.

This paper focuses on the class of scheduling problem in which:

1. activities have precedence relationships (one activity must not begin until another activity has completed);
2. resources are limited;
3. objectives or optimization criteria exist that may be used to rank competing schedules; and
4. the time frame in which to complete all activities (or as many activities as possible) is limited.

This class of problem differs from the job-shop problem domain in that a job-shop problem assumes an infinite time line in which all activities may complete. In a job-shop problem, all activities are scheduled regardless of the total time required. In contrast, in this paper resources may be over-subscribed, so that even the optimum schedule might not accommodate all desired activities within the time limitations. Thus, provision must be made for selecting between competing activities (or sets of related activities) where insufficient time exists for the completion of all activities.

To assist users in developing viable schedules, NASA has developed COMPASS (COMPuter Aided Scheduling System) [3], a computer-based tool that interactively schedules activities in a user-specified order. COMPASS provides graphical tools for displaying activities, resource availability, and schedules. An activity defined in COMPASS may have precedence requirements and require resources. Activity attributes supported by COMPASS include priority, required resources, duration, earliest permissible start time, latest permissible end time, and state conditions. COMPASS has enjoyed widespread acceptance and use within NASA and the contractor community.

NASA has recently proposed enhancing COMPASS to support multi-user or distributed scheduling problems. This paper focuses on the issues raised by distributed scheduling and on requirements for computerized support of this problem domain. The next section of this paper defines distributed scheduling and addresses these issues. This is followed by a discussion of some of the human issues involved in the development of protocols for use by multiple teams of schedulers who must cooperate to produce joint schedules.

DISTRIBUTED SCHEDULING

Definition

Distributed scheduling consists of those scheduling problems involving:

1. several schedulers,
2. who can work independently,
3. each of whom is responsible for scheduling separate sets of activities that are somehow interrelated, and
4. must share a common pool of resources.

Besides sharing a common resource pool, the activities may also have precedence requirements, or one activity may establish a state that another activity requires, etc. While the schedulers may work independently, the need to coordinate the interactions among their tasks prohibits purely independent work. Distributed scheduling problem domains of particular interest to NASA include the scheduling of astronomical satellite experiments, personnel training, and space mission activities.

The interactions among schedulers can be cooperative or competitive. Cooperative scheduling is defined as those cases in which:

1. All schedulers have the same objectives;
2. Responsibility for scheduling has been divided in order to share the labor; and
3. Protocols serve primarily to coordinate and synchronize.

In large problems the size and complexity of the scheduling task and the limited abilities, skills, knowledge, and resources of any individual make the distribution of the scheduling task a natural and necessary means of developing the required schedule. By distributing the work, each scheduler can concentrate on a manageable volume of work in a narrow domain. Some schedulers may develop specialized knowledge and skills suitable only to their particular domains.

In contrast, competitive scheduling consists of those cases in which:

1. Each scheduler has his or her own objectives;
2. The necessity of sharing common resources interferes with the simultaneous achievement of these objectives;
3. The pursuit of individual objectives leads to competition for the common resources; and
4. Protocols serve largely to arbitrate competition by allowing all schedulers fair access to shared resources.

Competitive scheduling can arise in situations in which there are contractual agreements among different parties or in which different resources are owned by different groups. Such situations can dictate the division of responsibility for scheduling among several groups, with each group having its own set of goals.

General Discussion of Goals

The lack of a single point of control increases the complexity of the overall scheduling problem (as a result of the necessary communication overhead) and raises several issues regarding the interactions of multiple schedulers and the integration of their individual schedules. The most basic issue raised by distributed scheduling is that of goals. What measure of goodness is most appropriate in a distributed environment? How do the optimization criteria for a distributed scheduling problem differ from those for a non-distributed problem? Variables commonly used for scheduling problems include [4, 5]:

-- Completion time: the time at which processing of the last activity completes.
-- Flow-time: the total time that activities spend in the shop.
-- Lateness: the difference between the completion time of an activity and some pre-specified due date associated with that activity.
-- Tardiness: equal to lateness when lateness is positive, otherwise equal to zero.

Schedule evaluation criteria typically involve minimizing or maximizing the mean, total, minimum, or maximum of one or more of these variables. In a standard job-shop problem, these criteria are assumed to be universally agreed upon. However, even in such a standard, non-distributed scheduling environment, the various tasks to be scheduled may belong to several different customers (perhaps represented by members of the marketing staff), each of whom would prefer that his or her tasks be given high priority. Thus, even in a non-distributed setting conflicting goals may exist. When conflict exists, the scheduler must have some means of determining a set of priorities to be applied to the scheduling task. The scheduler may be flexible in his or her choice of priorities, adjusting them to the needs of the moment. For example, the scheduler might attempt to mollify a major customer who has previously been slighted by giving preference to that customer's work in the current schedule. Regardless of the conflicting demands, however, the optimization requirements are formulated under a single point of control and this procedure can succeed because the single scheduler (or team of schedulers) who develops the optimization criteria also controls the entire resource pool.

In a distributed schedule, however, individual schedulers must share resources, so one scheduler optimizing his or her schedule may restrict another scheduler's options, resulting in a suboptimal global schedule. The issue of a global measure of goodness becomes more important in distributed scheduling than in individual scheduling. This is true because an individual scheduler can accept a schedule even without a specific measure of goodness; the schedule may balance several conflicting needs fairly and "just look good." A distributed schedule, in contrast, must "look good" through several sets of eyes. When a team of schedulers must continue to work together on future projects, perceptions of inequity or misplaced priorities can engender resentments that will poison these on-going relationships. Thus, some mechanism for balancing both local and global optimization must be provided. The protocol used by the schedulers to coordinate their activities must support optimization techniques that are perceived as both equitable and efficient.

Requirements for Competitive Scheduling

NASA needs to develop protocols that facilitate the development of successful schedules in "competitive" distributed environments that generally satisfy the objectives of the separate schedulers. This requires protocols that govern the process of building the schedule as well as protocols that govern how conflicting objectives are resolved. Selecting a desirable scheduling protocol requires balancing several possibly conflicting requirements, including the following [1]:

1. The protocol should encourage the development of high quality schedules that score well when evaluated by either the global optimization criteria or the optimization criteria of individual schedulers. Where conflicting objectives exist, the protocol should lead to a reasonable compromise.
2. The protocol should be easy to understand, use, and implement. Features enhancing ease of use include ease of learning; minimum complexity; informative to the user of the state of activities, resources, etc.; and natural representation of concepts. Yet the process should be sufficiently rich in features and notation to encompass a wide range of scheduling problems.
3. The protocol should be mechanical and unambiguous.
4. The protocol should be general enough to work with a wide range of scheduling software. Schedulers should not be constrained to use a particular scheduling system or even the same system.
5. The resulting schedules should be resilient to unexpected changes.
6. The overhead should be kept to a minimum. For example, the volume and frequency of communications should be low.
7. The time required to develop schedules should be short, especially in highly dynamic environments.
8. Any computerized support should have a short response time. This requires that optimization techniques be computationally simple.
9. Rescheduling (the repair of a schedule because of unexpected occurrences, such as delays and loss of resources) must be especially fast.

Several sample scheduling techniques are listed below. The alternatives are discussed in terms of division of resources, communication, cooperation, and optimality.

1. Schedule tasks by priority. This approach requires that all tasks be known and prioritized in advance and then be scheduled in priority sequence. This is really non-distributed scheduling, except that we have several schedulers responsible for collecting tasks and we may provide improved computer support to enable the individual schedulers to track their own set of tasks by viewing only their portion of the schedule. This protocol also requires some mechanism for assigning priorities to tasks, such as a central authority or a voting scheme. (Schedulers with conflicting objectives may never agree on the assignment of priorities.) Although participants may perceive this method as fair (since no lower priority

activity will be scheduled while a higher priority activity remains unscheduled), following this method strictly does not allow for compromises, such as scheduling two medium priority, low resource intensive activities instead of one higher priority, high resource intensive activity.

2. First come, first served. In this approach all schedulers are equal and none has priority over the others. Resources are not assigned to individual schedulers, but may be reserved by any scheduler. No cooperation among schedulers is required. Optimization is poor, because no attempt is made to balance the needs of multiple schedulers. There is a tendency among schedulers to reserve resources early, even before they know their full requirements. This hoarding can result in the allocation of resources to low priority tasks.
3. Divide resources among schedulers in advance. This method permanently allocates resources to specific schedulers who can use them as they choose. No communication or cooperation among schedulers is required. Schedulers need not even know the global schedule. This approach is impractical when there is a potential state conflict between tasks (e.g., when two schedulers independently schedule a treadmill experiment and a microgravity experiment that requires no vibration). This approach may also yield poor schedules when one scheduler assigns resources to low priority tasks or leaves resources unused that could be used by another scheduler. In this approach the quality of the resulting schedule is limited by the appropriateness of the initial allocation of resources. A poor allocation may result in few activities being successfully scheduled.
4. Divide resources among schedulers in advance but permit borrowing. This approach differs from the previous one by permitting schedulers to negotiate among themselves to improve their schedules. There is still no need for global optimization criteria. The status and bargaining power of individual schedulers is determined by the initial allocation of resources. Communication needs consist of a knowledge of resources available to other schedulers.
5. Sharing of intentions among schedulers. In this approach schedulers review their intentions with their peers and receive feedback before reserving resources and committing to a particular schedule. While this approach has the potential for producing high quality schedules through the sharing of knowledge and expertise, it also imposes a heavy communication burden among schedulers that can negate much of the benefit resulting from distributing the scheduling task. This approach is also fragile in that its success depends on the voluntary cooperation of each scheduler. Where this cooperation fails, this approach can degenerate into a first come, first served system.
6. Simultaneous iterative scheduling. In this method each scheduler devises a schedule and shares it with others. Schedulers identify and resolve conflicts by some agreed upon method. If unscheduled tasks and unallocated resources remain, another round of scheduling follows. In this approach all schedulers must be ready to schedule simultaneously.

Also, each participant must be provided some incentive to cooperate with the others in resolving conflicts. The global schedule must be available to all schedulers.

7. Consecutive iterative scheduling. In this method the schedulers are divided into two or more groups that alternately devise schedules. This approach is useful when one group creates resources required by another. For example, a university administration develops a schedule of classes, the students then submit their individual schedule requests, and the administration, after analyzing the requests, adds sections to some classes and deletes sections from others. The students then request changes to their schedules. In principle this cycle can continue for many iterations. This approach requires some incentive to cooperate and requires that each scheduler knows the global schedule and the state of available resources.

Any attempt to develop a universal scheduling methodology is doomed to failure because of the enormous diversity of scheduling domains. The variety of tasks, resources, constraints, and environments is virtually unlimited. The methodologies listed above are not applicable to all domains but must be selected based on the characteristics of the specific domain of interest.

Several other issues that are particularly relevant to distributed scheduling are briefly addressed in the remainder of this section. One of these is the requirement for revising a schedule, also termed rescheduling [2]. Several factors can trigger a need to reschedule. A resource can become unavailable, making the current schedule unfeasible; a task can take longer than expected; or a user can change his or her requirements so as to impose a conflict, exhaust a resource needed by a later task, or delete an enabling task that creates a subsequently needed resource or state. In addition, rescheduling is desirable, although not required, whenever an opportunity arises to improve the schedule by adding previously unscheduled tasks or resequencing already scheduled tasks. This can happen, for example, when new resources become available or when a task completes early. Differences between scheduling and rescheduling include:

1. Rescheduling takes place in the context of an existing schedule that we may wish to disturb as little as possible;
2. Rescheduling must consider work in progress;
3. Rescheduling often must occur quickly, in contrast to the initial scheduling which may be performed in a more leisurely manner; and
4. Someone other than the original scheduler may perform the rescheduling.

An important issue for rescheduling in a distributed scheduling environment is the need to reduce communication requirements among schedulers to facilitate quick rescheduling. Since this may require a return to centralized scheduling, the rescheduler must have the appropriate information to make beneficial changes.

Another issue is that of database support for distributed scheduling. A distributed scheduling system requires many of the features of a distributed database management system. The system must merge separate databases of tasks, resources, constraints, and assignments into a single image

while retaining the ability to display for individual schedulers only those portions of the database under their control. However, since each scheduler has a different view of the world (with different granularity levels, time scales, measures of goodness, types of constraints, etc.), the system must support different user languages and communicate with each scheduler in a natural and helpful way. As our software tools, such as COMPASS, address more diverse and complex problem domains, we will require a more comprehensive database language for describing scheduling problems.

Communication and coordination among schedulers in an important issue. NASA schedulers who impact one another may work at different centers, making communication difficult. The scheduling of Space Station Freedom will involve groups in several countries. An important research question concerns how frequently NASA schedulers communicate. Is the level of communication optimal? If it is below optimal, do schedulers fail to communicate because they do not perceive a need to communicate, or because they feel communication is too time consuming, or because they fear loss of control of their environment, or is there some other reason? If independent scheduling is a human preferred approach, then it will be important to determine why this is true, how we can encourage people to cooperate, and how we can enhance cooperation while minimizing communication. The mechanisms for communication and coordination (the languages, database support, and interaction procedures) appear to be a critical aspect of distributed scheduling by human agents.

A final issue involves the introduction of expert system support for scheduling. Optimization heuristics have been envisioned for individual scheduling support; some of this support is already available on COMPASS. Expert system support for distributed scheduling would focus on communication and negotiation. An expert system that monitored the actions of all schedulers could infer when one scheduler needed to know of the actions of another. This technique could reduce communications requirements among human schedulers. Also, an expert system could search for instances in which two schedulers could trade resources or reschedule certain activities to their mutual advantage. Ultimately, we may wish to introduce artificially intelligent schedulers into a distributed scheduling system. The scheduling of certain domains, such as power generation, may be suitable for AI approaches. Once AI schedulers are developed for individual scheduling domains the natural next step would be to introduce them into human scheduling systems. This possibility raises questions regarding how artificial and human schedulers might best interact.

SAMPLE PROTOCOL: THE RED-BLUE PROTOCOL

The Red-Blue protocol has been devised to guide the interactions between schedulers at NASA/JSC and SpaceHab, Inc. of Huntsville, Alabama as they schedule STS-57, due to launch in April, 1993. The objective of this protocol is to facilitate the production of payload deployment and management schedules in the context of other orbiter/station operations. Based on our experience with scheduling this mission, the Red-Blue protocol will be enhanced and used as NASA's standard protocol for the distributed scheduling of shuttle (and, later, space station) operations.

The requirements for the Red-Blue protocol are similar to those discussed above. It should be easy to understand, use, and implement. Its use should be mechanical and unambiguous. It should work with a variety of scheduling software systems. It should support rescheduling. Its requirements for communication among schedulers, in terms of frequency and volume of communication, should be low. It should allow the creation of schedules in a timely manner. Finally, where there are conflicting objectives, the protocol should lead to the creation of schedules that provide a reasonable compromise between these objectives.

The Red-Blue protocol begins by dividing all of the activities into two groups, red and blue. The red activities can only be scheduled by the red scheduler, in this case, NASA. Likewise, the blue activities can only be scheduled by the blue scheduler, SpaceHab, Inc. Limits can be placed on the volume of resources that can be used by the red and blue schedulers; for example, a scheduler might be limited to a maximum quantity of water during the mission or a maximum number of hours of an astronaut's time. Limits are not placed, however, on where in the schedule the resources may be used. In the case of NASA and SpaceHab, Inc., these limits have been established during contract negotiations. Any subsequent modifications or clarifications to these limits must be worked out by the schedulers and possibly their management.

The red scheduler produces the first schedule, placing red activities anywhere on the timeline, up to the limit of the red resource allocation. For example, the red scheduler would schedule basic activities such as course correction burns and astronaut sleep and meal times as well as mission specific activities such as payload deployment. Next the blue scheduler places blue activities in any available (white) space on the timeline, up to the limit of the blue resource allocation. Thereafter, only one scheduler may work on the timeline at a time. The scheduler who has authority to modify the schedule at any particular time is said to "hold the token." When the other scheduler has activities to schedule, that scheduler may request the token. The scheduler who holds the token may schedule or move his or her activities within any white space and within any space that he or she already occupies. The scheduler may not, however, move any activities of the other scheduler, or oversubscribe any resource.

If one scheduler wants to move an activity into the space owned by the other scheduler, the two schedulers can negotiate a set of changes that can then be produced by operations according to the basic protocol. While this protocol assumes that the parties are competitive (having differing and possibly conflicting goals), it also assumes that they are not antagonistic. Thus, the protocol assumes that the parties will cooperate whenever the result of such cooperation leaves neither party worse off.

A low communication procedure for asking the other party to move some of its activities is to allow a scheduler to oversubscribe resources (thereby producing a conflict between red and blue activities). The other scheduler, when he or she next holds the token, can leave the oversubscription (thereby delaying the resolution of the conflict), unschedule the offending activities of the other color, or accommodate his or her counterpart by moving some activities of his or her own color.

If more than two schedulers need to work cooperatively, then the Red-Blue protocol can be extended by devising a procedure for exchanging authority to operate on the schedule. A research question is to investigate the social and communications changes that occur as the number of scheduling groups rises.

Several implementation issues must yet be addressed. When activities must be rescheduled during a mission, does one party have the right to force the other to modify its schedule? For example, if an activity runs long, can the other scheduler force termination of the activity? Also, electronic protocols must be developed for the exchange of schedule updates. These protocols must enforce the requirement that only the token-holder may modify the schedule and must ensure that all parties always agree on the composition of the current schedule. Thus, when one party refers to the current schedule or to the schedule as it existed two versions ago, the other party will know what is meant.

CONCLUSION

NASA has an unlimited variety of distributed scheduling problems. Competitive distributed scheduling problems arise when shared use of common resources interferes with the simultaneous achievement of multiple resources. We need to develop protocols that govern the process of building the schedule and govern how conflicting objectives are resolved. These protocols can only be developed and evaluated in the context of specific applications. This paper presents a simple, yet effective Red-Blue protocol to facilitate the production of payload schedules in the context of other orbiter/station operations.

REFERENCES

[1] Fox, Mark S., "Constraint-Directed Search," Ph.D. dissertation, Carnegie-Mellon University, 1983.

[2] Fox, Mark S. and Zweben, Monte, "Knowledge Based Scheduling," Tuturial MA2, presented at the Ninth National Conference on Artificial Intelligence, July 15, 1991.

[3] McDonnell Douglas Space Systems Co. "COMPASS 2.0 User's Manual," for NASA/Johnson Space Center Software Technology Branch, 1991.

[4] Ow, Peng Si, "Heuristic Knowledge and Search for Scheduling," Ph.D. dissertation, Carnegie-Mellon University, 1984.

[5] Salvador, Michael S., "Scheduling and Sequencing," in *Handbook of Operations Research*, Joseph J. Moder and Salah E. Elmaghraby (Eds.), Van Nostrand Reinhold, New York, 1978, pp. 268-300.

[6] Ullman, J. D., "NP-Complete Scheduling Problems," in *Journal of Computer and System Sciences*, Vol. 10, 1975, pp. 384-393.

SHUTTLE GROUND PROCESSING SCHEDULING

Mike Deal
KSC
L50-459
1100 Lockheed Way
Titusville, FL 32780

Abstract unavailable at time of publication.

Scheduling with Partial Orders and a Causal Model

Mark Boddy Jim Carciofini George D. Hadden
{boddy | carciofi | hadden}@src.honeywell.com
Honeywell Systems & Research Center, MN65-2100
3660 Technology Drive
Minneapolis, MN 55418

Abstract

In an ongoing project at Honeywell SRC, we are constructing a prototype scheduling system for a NASA domain using the "Time Map Manager" (TMM). TMM representations are flexible enough to permit the representation of precedence constraints, metric constraints between activities, and constraints relative to a variety of references (e.g., Mission Elapsed Time vs. Mission Day). The TMM also supports a simple form of causal reasoning (projection), dynamic database updates, and monitoring specified database properties as changes occur over time.

The greatest apparent advantage to using the TMM is the flexibility added to the scheduling process: schedules are constructed by a process of "iterative refinement," in which scheduling decisions correspond to constraining an activity either with respect to another activity or with respect to some timeline. The schedule becomes more detailed as activities and constraints are added. Undoing a scheduling decision means removing a constraint, not removing an activity from a specified place on the timeline. For example, we can move an activity around on the timeline by deleting constraints and adding new ones, and other activities constrained with respect to the one we move will move as well.

1 Introduction

We are interested in the solution of large, complex scheduling problems. Examples of the kinds of domains we are interested in include several NASA scheduling problems (e.g. Spacelab, Space Station operations, Shuttle ground processing), and the *Transportation Planning* problem being addressed by the joint DARPA/Air Force Planning Initiative.

A "solution" as we use the term is not simply an implementation of an algorithm for solving a particular constraint satisfaction or constrained optimization problem. For many domains, constructing schedules is an extended, iterated process that may involve negotiation among competing agents or organizations, scheduling choices

made for reasons not easily implementable in an automatic scheduler, and last-minute changes when events do not go as expected. In such an enviroment, the process by which a schedule is constructed must be considered in any attempt to provide a useful scheduler for a given domain.

Even the more limited problem of generating a single schedule is becoming increasingly complex. Simple models solvable by straightforward application of standard operations research techniques such as linear programming are less and less relevant to current scheduling problems. For example, many NASA scheduling domains involve large problem instances (hundreds to thousands of activities and constraints), context-dependent activity effects (including context-dependent transitions or setup times as a special case), complex resource structures (e.g., a power bus that is divided into sub-busses), and user preferences on where activities appear in the final schedule (e.g., "as late as possible"). To provide an effective solution, a scheduling system must be expressive enough to represent or reflect these domain complexities as well as supporting the process by which a schedule is constructed.

In an ongoing project at Honeywell SRC, we are implementing a prototype scheduling system for a NASA domain using the "Time Map Manager" (TMM). TMM representations permit the expression of precedence constraints, metric constraints between activities, and constraints relative to a variety of references (e.g. Mission Elapsed Time vs. Mission Day). The TMM also supports causal reasoning (projection and persistence), dynamic database updates, and monitoring certain database properties as changes occur over time.

The greatest apparent advantage to using the TMM is the flexibility added to the scheduling process: schedules are constructed by a process of "iterative refinement," in which scheduling decisions correspond to constraining an activity either with respect to another activity or with respect to some timeline. The schedule becomes more detailed as activities and constraints are added. Undoing a scheduling decision means removing a constraint, not removing an activity from a specified place on the timeline. For example, we can move an activity around on the timeline by deleting constraints and adding new ones, and other activities constrained with respect to the one we move will move as well.

In the rest of this paper, we provide a brief introduction to the TMM, describe the application of the TMM to scheduling, and describe some related work.

2 TMM Overview

As part of the DARPA/RL Planning Initiative, Honeywell has developed a new implementation of Dean's Time Map Manager (TMM) [5], involving improvements in robustness, user interface, and documentation, in addition to a number of extensions in functionality. The TMM provides users and application programs (e.g., planners and schedulers) with the following functionality:

- Metric and ordering constraints between any two points.
- Causal reasoning.

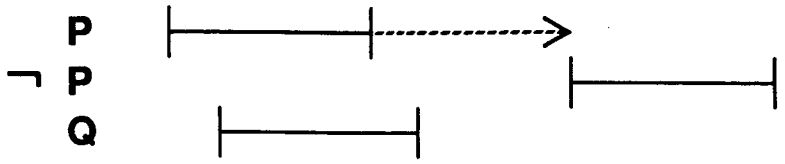


Figure 1: A simple temporal database

- Database monitors for temporal conditions and protections.
- Optimizations for large temporal databases.

The structure and capabilities of the TMM are described in more detail below.

2.1 Temporal Relations

The TMM lets users assert *constraints* between pairs of *time points*, resulting in a partial ordering among the points. TMM supports queries regarding necessary and possible temporal relations among the time points. The truth of facts over intervals of time is represented by *tokens*, which may include properties of *persistence* beyond their observed endpoints. In the current implementation, tokens may persist both forward and backward in time. The truth of a proposition over an interval is determined based on the ordering of token endpoints and the token's persistence properties. For example, Figure 1 is a simple temporal database, involving three tokens of three different types. In this example, P is true over the interval bounded by the vertical lines, and persists into the future. (not P) becomes true at a later time, and *clips* the forward persistence of P. The statement "P and Q" is true for an interval defined by the overlap of the tokens labelled P and Q.[1]

2.2 Causal Reasoning

The TMM supports reasoning about the changing state of the world as activities occur using three forms of inference:

- The persistence assumption. As described above, users of the TMM specify that certain facts are believed to be true over specific intervals of time. In addition, they can specify that those facts can be assumed to remain true until something occurs to make them false.
- Projection. This is inference of the form: given an event E and a set of preconditions $P_1, P_2, \ldots P_k$, and a result R, whenever the preconditions are believed to be true for the entire event E, R is believed to become true immediately following E.

[1] We are in the process of developing a formal semantics for the TMM. A draft version is available by request.

- Overlap chaining. Given a set of preconditions $P_1, P_2, \ldots P_k$, and a result R, R is believed to be true for any interval for which all of the preconditions are true.

All of these forms of inference are handled completely automatically: the user specifies which facts are persistent and asserts a set of projection and overlap rules, and the requisite inference is performed by the system.

2.3 Nonmonotonic Reasoning and Database Monitors

TMM supports two basic kinds of nonmonotonic reasoning:

- Possibly true temporal relations between time points (which may be invalidated by additional constraints), and
- Assumed truth of a temporal proposition over an interval based on a time token's persistence (which may be invalidated by the addition of a contradictory token, which *clips* the proposition during that interval).

In addition, the database itself is "nonmonotonic", in the sense that information can be deleted, and the inference performed by the system thus far will be checked to ensure that it continues to be supported by the current state of the database. [2]

The existence of specified database properties as changes are made over time can be tracked through the use of *monitors*. The existing types of TMM database monitors are *temporal conditions* and *protections*. Temporal conditions monitor whether specified relations among points can be derived from the current state of the database, maintaining this information as the database changes. Protections do the same thing for the truth of some fact over an interval. Between them, these two mechanisms provide support for monitoring the continued validity of previous inference, or triggering demons based on complex properties of the temporal database

2.4 Efficiency

Current and planned TMM optimizations for handling large databases include the use of a global reference point where appropriate (rather than forcing its use as some systems do), limiting search to that necessary to prove or disprove a query, caching search results for later use, graph decomposition, temporal indexing, lazy monitor evaluation, and algorithms that are designed to search only those parts of the database that may result in useful answers.

3 Scheduling Using the TMM

The assumptions underlying our scheduling work are as follows:

[2] This capability (*temporal reason maintenance*) is described in detail elsewhere [5].

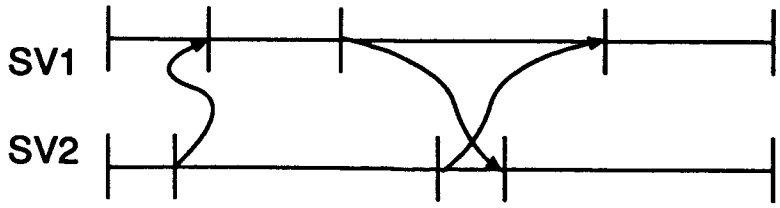


Figure 2: Time-line scheduling

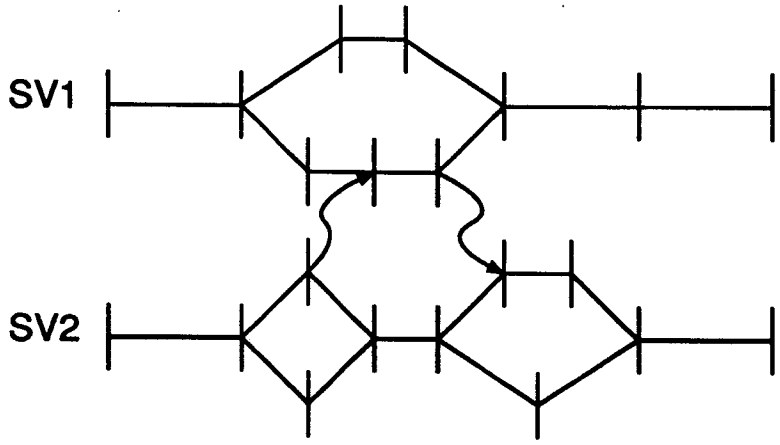


Figure 3: Constraint-posting scheduling and the resulting partial order

1. Explicitly modelling the constraints resulting from specific scheduling decisions makes the schedule easier to construct and modify.

2. Representing only those relationships required by the current set of constraints (the decisions made so far) provides a more useful picture of the current state of the scheduling effort.

The main consequence of this approach is that the scheduler does not manipulate totally-ordered timelines of activities and resource utilization. Instead, the evolving schedule consists of a partially ordered set of activities, becoming increasing ordered as additional constraints are added (or less so, as those decisions are rescinded).

Timeline schedules can be represented using linear sequences of tokens, one sequence for each resource. Figure 2 depicts a simple timeline schedule Arrows between the sequences represent constraints on parts of the two sequences that must obey the indicated ordering relationship. In contrast, schedules constructed by accumulating constraints have a structure like that in Figure 3. Here, the current set of constraints is insufficient to force a totally-ordered sequence of activities. Although providing increased flexibility (through delaying commitment), the explicit representation of partially-ordered activities in the time map makes reasoning about resource usage and other state changes more complicated. It is no longer possible to construct a single time-line representing (e.g.) changing resource availability over time. Instead, the system computes *bounds* on the system's behavior.

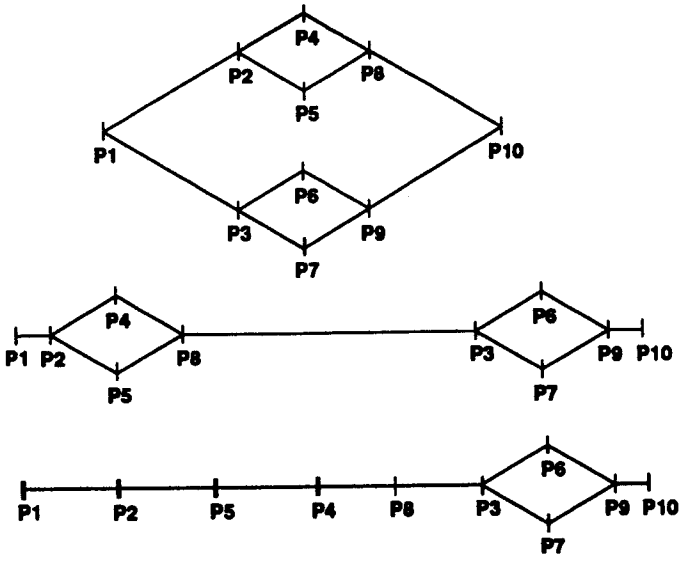


Figure 4: Gradual hardening of a partial order

Causal reasoning and resource profiles both depend on precise orderings of facts and activities in time, that is, on what propositions are true and what activities occur when. For a partial order, we can determine what facts might *possibly* or *necessarily* hold at a point, in some or all of the total orders consistent with the given partial order. With even a very simple causal model, this is an NP-complete problem [4]. The solution we have implemented (first presented in [3]) is to approximate the necessary quantification, implementing *strong* and *weak* reasoning as approximations for what is possibly or necessarily true, given the current partial order. [3]

Figure 4 depicts the process by which a partially ordered schedule is gradually refined into an executable, totally ordered schedule. In our approach to constructing the final schedule, this is one of the ways in which a partial or incomplete schedule is modified, the other being the addition of new activities.

The TMM's strong and weak reasoning provides a partial solution to the problem of reasoning about what will happen. For certain classes of inference, in particular problems involving resource capacity or the aggregate duration of mutually exclusive activities, strong and weak (even exact "necessary" and "possible") reasoning occasionally provides insufficient information. For these cases, there are two possible approaches: simulation (sampling) of totally-ordered sequences, or some kind of static graph analysis to determine better bounds on the system's behavior. The end result in either case is a measure of how likely it is that further constraints on the partial order will cause problems, requiring the scheduler to backtrack to earlier choices.

Despite the approximate nature of this reasoning, we are still ahead of the game: where the least-commitment approach to scheduling can at least provide

[3] See [5], or [13] for details.

approximate answers in support of scheduling decisions (e.g. what order activities should occur in), timeline schedulers make the same decisions arbitrarily—putting an activity on the timeline is a stronger commitment than constraining it to occur (say) between two other activities, or within a given time window.

4 Related Work

The idea that schedules should be constructed "from the side," looking at part or all of the schedule history rather that just sweeping forward or backward in time, has been implemented in several scheduling systems, e.g. [7, 18, 1, 14]. Typically, these systems also support an iterative process of schedule refinement or repair. Recent work on COMPASS provides a protocol for allowing different agents to modify the same schedule, wherein commitments made by one agent cannot be affected by the actions of any other. Research in constraint-based scheduling [8, 15, 12] has demonstrated the advantages of considering the structure of problem constraints over time and using this structure to dynamically focus decision-making on the most critical decisions. However, these systems have historically had a weak model of the interaction of activities and the evolving state of the domain.

Research in generative planning has focused on the construction of activity networks that bring about desired goal states, given basic representations of the effects of actions in the world. Classical domain modeling assumptions [6] [17] make it difficult to reason about the duration of activities, continuously varying quantities, and resource consumption. The consequence of these limitations is that automatic planners have not had any great success in applications to significant planning and scheduling problems [2, 16].

In addition to the TMM, several other temporal database systems have been implemented with an ability to reason about time "from the side" [11, 9, 10]. To date, the TMM's combination of expressive flexibility, precise semantics, support for database operations, and extensive optimization set it apart from other temporal reasoning systems.

5 Summary

Effective support for current scheduling domains requires a focus on the scheduling process as well as the scheduling problem. Using the TMM, we are in the process of constructing a novel form of scheduler that constructs schedules through the accumulation of constraints on the relations between activities, and between activities and various reference points (e.g. calendar time, mission elapsed time, etc.).

The TMM's support for flexible temporal relations, dynamic updates in large databases, and causal reasoning provide an effective base for building schedulers for complex problems. The TMM is freely available and written in Common Lisp. To obtain a copy of the software or a more detailed description of the system's design and capabilities, contact the authors at the address given on the first page.

References

[1] Biefeld, E. and Cooper, L., Scheduling with Chronology-Directed Search, *Proc. AIAA Computers in Aerospace VII, Monterey, California*, 1989, 1078–1087.

[2] Dean, T., Firby, R.J., and Miller, D., Hierarchical Planning Involving Deadlines, Travel Time, and Resources, *Computational Intelligence*, 4 (1988) 381–398.

[3] Dean, Thomas and Boddy, Mark, Incremental Causal Reasoning, *Proceedings AAAI-87 Sixth National Conference on Artificial Intelligence*, 1987, 196–201.

[4] Dean, Thomas and Boddy, Mark, Reasoning about Partially Ordered Events, *Artificial Intelligence*, **36**(3) (1988) 375–399.

[5] Dean, Thomas and McDermott, Drew V., Temporal Data Base Management, *Artificial Intelligence*, **32** (1987) 1–55.

[6] Fikes, R.E., Hart, P.E., and Nilsson, N.J., Learning and Executing Generalised Robot Plans, *Artificial Intelligence*, **3** (1972) 251–288.

[7] Fox, B. R., Non-Chronological Scheduling, *Proc. AI, Simulation, and Planning in High Autonomy Systems, University of Arizona*, IEEE Computer Society Press, 1990.

[8] Fox, M.S. and Smith, S.F., ISIS: A Knowledge-Based System for Factory Scheduling, *Expert Systems*, 1(1) (1984) 25–49.

[9] Ghallab, M. and Alaoui, A.M., Managing Efficiently Temporal Relations through Indexed Spanning Trees, *Proceedings IJCAI 11, Detroit, Michigan*, IJCAI, 1989, 1297–1303.

[10] Koomen, J.A.G.M., *The Timelogic Temporal Reasoning System*, Technical Report 231 (revised), University of Rochester Computer Science Department, October 1988.

[11] Materne, S. and Hertsberg, J., *MTMM: Correcting and Extending Time Map Management*, Technical Report 511, GMD, February 1991.

[12] Sadeh, N. and Fox, M.S., Variable and Value Ordering Heuristics for Activity-based Job-shop Scheduling, *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, Hilton Head Island, S.C.*, 1990.

[13] Schrag, Robert, Boddy, Mark, and Carciofini, Jim, Handling Disjunction in Practical Temporal Reasoning, *KR-92*, 1992.

[14] Smith, S.F., Ow, P.S., LePape, C., McLaren, B. S., and Muscettola, N., Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans, *Proceedings of the SME Conference on AI in Manufacturing, Long Beach, CA*, 1986.

[15] Smith, S.F., Ow, P.S., Potvin, J.Y., Muscettola, N., , and Matthys, D., An Integrated Framework for Generating and Revising Factory Schedules, *Journal of the Operational Research Society*, 41(6) (1990) 539–552.

[16] Vere, S., Planning in Time: Windows and Durations for Activities and Goals, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5 (1983).

[17] Wilkins, D.E., *Practical Planning*, volume 4, (Morgan Kaufmann, 1988).

[18] Zweben, M., Deale, M., and Gargan, R., Anytime Rescheduling, *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control, San Diego*, DARPA, 1990.

Crisis Action Planning and Replanning using SIPE-2

Jennifer D. Skidmore
Rome Laboratory
RL/C3CA
Griffiss AFB, NY 13441-4505
skiddles@aivax.rl.af.mil

Abstract

Rome Laboratory and DARPA are jointly sponsoring an initiative to develop the next generation of AI planning and scheduling technology focused on military operations planning, especially for crisis situations. SRI International has demonstrated their knowledge-based planning technology in this domain with a system called SOCAP, System for Operations Crisis Action Planning. SOCAP's underlying power comes from SIPE-2, a hierarchical, domain-independent, non-linear AI planner also developed at SRI. This paper discusses the features of SIPE-2 that made it an ideal choice for military operations planning, and which contributed greatly to SOCAP's success.

1 Introduction

The goal of the DARPA/Rome Lab Planning Initiative is to develop the next generation of artificial intelligence planning and scheduling tools focused on military operations planning, especially crisis action planning. Developing an appropriate course of action (COA) in response to a crisis situation encompasses the planning of both the *employment* of forces against the enemy and the *deployment* of forces and cargo to their required destinations in time to complete their mission.

This planning is done at various levels along the chain of command. Specifically, when a crisis occurs that requires military action, the Chairman of the Joint Chiefs of Staff passes down a mission statement with some planning guidance to the Commander-in-Chief (CINC) of the appropriate geographical area. The CINC and his staff are responsible for generating alternative COAs based on available intelligence data about the enemy's posture, terrain information, approaches that have proven successful in the past, logistics capabilities, and many other factors. The CINC is mostly concerned with the employment plan and whether it can win the war. It is then the job of the U.S. Transportation Command (USTRANSCOM) to determine how to get the CINC's required forces, equipment, and support units into theater by the time they are needed. The war-fighting CINC requires feedback if it is determined that his employment plan is transportationally infeasible. If not enough transportation resources are available to meet the deadlines, then some replanning may need to be done on the employment end, priorities may need to be shifted, or more resources may need to be negotiated from the commercial sector. In a crisis situation, this feasibility analysis and replanning cycle needs to happen quickly, on the order of hours.

The first product of the Planning Initiative, DART (Dynamic Analysis and Replanning Tool), was built for USTRANSCOM to automate and speed up the editing and evaluation of deployment plans. These "plans" are called TPFDDs, and specify all the units being moved, including major forces, supporting forces, or resupply, the mode of transport for each unit (air, land, or sea), and the partial schedule of departure dates and arrival dates. DART supports the analysis of TPFDDs by passing the information through standard simulations and receiving feedback on late arrivals of people or cargo. Planners at USTRANSCOM are then able to use DART to highlight some of the problem areas in the TPFDD so that they may intelligently revise the TPFDD before analyzing it again. Operational users have been trained on the DART system, and they are using it to help them to their day-to-day jobs.

Currently, however, the CINC's staff has no automated support for initially generating plans for force employment or deployment; the plans are still built up by hand. The truth is that there is not a single generative planning system in operational use anywhere in the military today. The technology has so far been considered too immature for real world-sized applications. One of the goals of the Planning Initiative is to show that AI technology is ready to be integrated into decision aids for the military operations planning/replanning process. The first system to step up to the challenge is SRI International's System for Operations Crisis Action Planning (SOCAP), which incorporates the SIPE-2 planning system also developed at SRI.

The next section talks about the SOCAP system, and the rest of the paper focuses on the features of the SIPE-2 planner embedded in it that contributed to its success. Attention is also given to those aspects of SIPE-2 that caused problems, or are considered areas for future work.

2 SOCAP

SOCAP was built by SRI International to demonstrate the feasibility of applying the planning technology of SIPE-2 to the generation of large-scale military operations plans. For the purposes of a large demonstration at U.S. Central Command and the Pentagon in January 1992, SOCAP was connected to DART, with a module in between that was able to take the major forces from the SOCAP-generated plan, fill in the additional support forces, sustainment, and resupply, and generate a TPFDD. After DART was used to determine the feasibility of the plan, it was shown how changes made at a high-level in the SOCAP plan (as opposed to local changes in the TPFDD) could remedy the situation.

SOCAP embodies the domain-independent planner, SIPE-2, together with the domain knowledge necessary to apply SIPE-2 to military planning, plus a user interface designed especially for military planners which makes effective use of a map display system. Figure 1 shows the SOCAP architecture, highlighting the database inputs, the user inputs, and the outputs.

On the left are the inputs to SOCAP that would come from military databases, such as a list of enemy threats and their locations, combat forces available for the operation, and geographical location information on ports, bases, etc. The user would input the goals or missions to be accomplished, and any special constraints or assumptions that he wanted to the plan to take into consideration. In the main module in the middle of the figure is the SOCAP application layer around the SIPE-2 core.

Built into the SOCAP knowledge base are the objects of the domain, the operators which describe military operations used in a plan to achieve the goal(s), the constraints that need to be maintained, and the classes of resources needed by the plan operators. The objects and their properties (for example, the combat capabilities of forces or the capacities of ships and ports) are stored in SIPE-2's sort hierarchy, while more dynamic information is represented as predicates. Because SIPE-2 is a hierarchical planner, SOCAP has operators at all levels of abstraction, and the representation of an operator is consistent whether it is a primitive action that can be taken in the world, or a subgoal that still needs to be achieved. The size of the SOCAP knowledge base is given in [2]: 200-250 classes and objects, 15-20 properties per object, around 1200 predicates, and 50-100 plan operators.

Using all this information, SOCAP will generate a plan, either automatically, or with user interaction/guidance. The user can actually decide at each level of the planning hierarchy how much of the decision making he would like to do. It was to support this capability that a new operationally oriented interface (distinct from the SIPE-2 developer's interface) was developed. The SOCAP interface provides extra capabilities for guiding the user through planning process. If desired, SOCAP can display, at each goal in the plan, the list of operators that can be used to achieve that goal. Likewise, when choosing a particular value for a variable, such as which infantry brigade to use in an operation, the user may opt to be presented with a list of brigades that satisfy the constraints on the variable and choose one himself. At the end of each plan level, SIPE-2's plan critics are called to check that the plan decisions made so far are consistent.

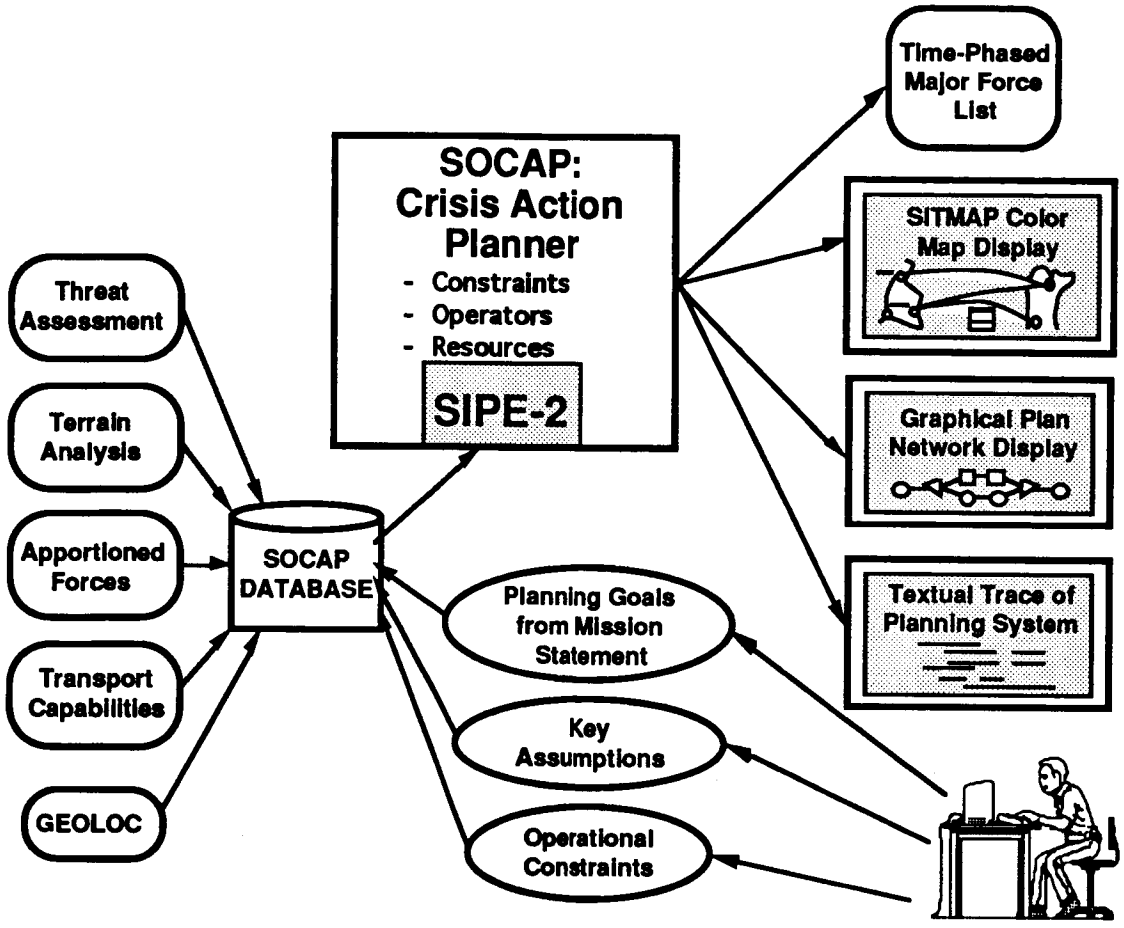


Figure 1

Usually, the user views the plan in the style of SIPE-2, as a network of partially-ordered plans and goals where nodes are actions and goals and arcs are ordering links. Additionally, SOCAP added the ability to view a developing plan on a map display that shows where the major forces are located on any day of the plan. This type of display is popular with military planners who think about war plans as arrows and circles on a map rather than as a graph.

When there is no more planning to be done, SOCAP outputs, in addition to the fully expanded plan network and map-based plan, a time-phased list of the major forces involved in the operation. This is what is then fleshed out to produce a TPFDD which can be shipped to the DART system for analysis.

3 SIPE-2

At the heart of the SOCAP system is SIPE-2, the artificial intelligence planner that understands how to build plans and can reason about the effects of actions taken in the world. SIPE-2 was chosen for this demonstration because it is a domain-independent planning system that has been successfully applied to several domains already, including an extended blocks world, mobile robot planning, office building construction, travel planning, and brewery production line scheduling. SOCAP added the domain-specific knowledge of the military operations planning world to the core planning engine of SIPE-2. As has been discussed briefly in the preceding section, this involves describing objects, actions with their

preconditions and effects, constraints, and resources. Also encoded in a SIPE-2 application are deductive operators that encode which changes in the world entail other changes and some domain-specific heuristics for improving the efficiency of planning.

It is significant that these things are fairly easy to represent in the SIPE-2 formalism because there is so much domain information that needs to be built in for each new scenario. The particular way in which the knowledge is built in can affect the quality of plans produced and the speed with which they are generated. For example, everything in the world could be entered into SIPE-2 as predicates; however, it is much more efficient to store the static properties of objects, such as the speed of a C-5 aircraft, in SIPE-2's sort hierarchy. This can either be viewed as an advantage or a disadvantage depending on whether the application developer is experienced with SIPE-2, or whether the user wants to add a new operator type or a new domain constraint. Some ideas have been discussed for future work to develop a user-friendly intelligent interface tool for the purpose of assisting a non-SIPE expert in extending the domain.

SIPE-2 is a hierarchical planner, which means that it has operators and goals at various levels of abstraction, and it expands all the goals at one level of a plan before trying to solve any subgoals. Planning continues until all subgoals have been achieved by actions, or until it is found that a goal cannot be satisfied. This hierarchical decomposition of a top-level goal is generally a very natural way to think about the problem-solving process. It also makes planners more efficient since there are fewer applicable operators to consider at each level, and backtracking can be reduced significantly by working out conflicts at an earlier level before all the details get filled in.

It was relatively easy to group SOCAP's plan operators into sets corresponding to the various phases or levels of command in the planning process. In the demonstration scenario, the top-level goal is to protect the territorial integrity of a third world country against its neighboring enemies. The first level of the plan is to select a mission type, such as show-of-force or deterrence. This would be the kind of decision that the JCS would make. The second level of the plan identifies the threats and their locations and sets up multiple subgoals to counter each of those threats. With the information about the threats, the war-fighting CINC would then decide on what kinds of forces to employ and how to employ them to counter the threats. Thus, level 3 of the SOCAP plan expands each of the previous subgoals into an employment action using a particular force or unit and a deployment subgoal for the unit to arrive at the destination of the threat in time. Actually this step should be broken up into two levels, since the CINC usually specifies only what type of force is required, and the "sourcing" or selecting of specific forces is done by FORSCOM. At this stage of the planning process, it is TRANSCOM's role to flesh out the deployment subgoals of the designated forces, along with supporting forces, and sustainment and resupply, given the available transportation assets. This is reflected in the plan that has developed after level 3, as the employment actions are all filled in and cannot be broken down further, but deployments are still shown as goals.

As you can see, the employment portion of the plan does not include much detail. But SIPE-2's hierarchical nature will make it easy to add in the lower-level employment operators in the future. As a demonstration, however, SOCAP was quite successful in exploiting the hierarchical planning capability of SIPE-2 to match the military planning process.

Besides complexity and speed (the lack thereof), one of the reasons that generative planners are not currently used in the military is that operational users want to be in charge of the planning process rather than letting a computer tell them how to conduct a military campaign. And rightfully so; humans have much valuable information and insight that is too subtle, or just too massive, for a computer to handle. But there is no reason why an intelligent computer can't share the burden.

SIPE has always been capable of both automatic and interactive planning. It can be so interactive, in fact, that SIPE's role becomes one of merely guiding the user through the process of building a plan, offering suggestions, and recording decisions made. The user can vary the level of interaction as desired during the planning process. However, SIPE-2 is so flexible about the interaction, what items can be displayed and what choices can be made, that it can be too confusing for a non-SIPE expert. SOCAP's

interface is domain-specific and extracts the information from SIPE-2 that a military planner cares about. For example, SIPE-2 hides a lot of constraint information, so when choosing a value for a variable on an operator, it might not be obvious which choices satisfy the constraints imposed by that operator. The builders of SOCAP decided the extract the constraint information right away and only present to the user the consistent choices.

SIPE-2 is also a non-linear planner, so it can represent actions that are unordered with respect to each other (and possibly simultaneous). By not forcing ordering links between actions, SIPE-2 reduces the amount of backtracking caused by an action's effects violating the preconditions of a later action. Also, for some plans you want to represent possibly simultaneous actions, where they can either be executed in parallel or it doesn't matter which is executed first. (It may be interesting to note that SIPE-2 does not have the temporal reasoning capability to make this distinction.) This non-linearity of plans was an extremely important feature for military operations plans, where several actions are being executed at the same time using sharable or completely different sets of resources. In the demonstration scenario, land, sea, and air forces are able to all work on their missions simultaneously, and many deployment actions also occur on the same day, simply using different resources.

This brings up the issue of SIPE-2's special handling of resources. SIPE was the first AI planner to allow resources for an operator to be specified explicitly. SIPE-2 has been enhanced to handle reusable and consumable resources. For example, a transport-by-sea operator, instead of having a precondition of "big enough ship available during time-period1", would have "ship" listed as a reusable resource, and SIPE-2's resource contention critics would check to make sure the resource was available at that time.

SIPE-2 also has mechanisms for handling sharable resources, but in designing SOCAP, they were found to be too inflexible. For instance, as Roberto Desimone writes in [2]:

> "a large military unit, such as a division, may be employed in several operations simultaneously, where each operation uses some of the division's capabilities. The number of operations over which the division may be shared depends of the amount of resources required for each operation. Thus, the only way to reason about the shared resource is to consider the capabilities of the division as a consumable resource purely for this specific set of operations."

Even the resource reasoning capabilities that SIPE-2 does embody were not used effectively in SOCAP because of the lack of temporal reasoning capability. SIPE-2 has no concept of how long an action takes, or of time intervals between actions. Therefore, sometimes it appears that there is a resource conflict between unordered actions when, if there was only information about the times during which a resource was unavailable or about the time intervals of the actions, there might not be a problem. In SOCAP, there are dates associated with actions in the plan, but these numbers are not used at all by SIPE-2, and one may find two operations, one with a date of 24 and one with a date of 26, that are unordered with respect to each other in the plan.

One of the big wins for SOCAP is SIPE-2's powerful constraint representation language, which is richer than that of most planning systems. The ability to post constraints on the variables of plan operators as they are added to the plan can save a lot of time over the method of choosing a binding for the variable right away. The variable binding decision can sometimes be delayed until the constraints posted and propagated to it point to a single value. Also, having declarative constraints in the system aids the user interface in generating explanations for why a particular planning decision doesn't apply.

One of the difficulties with the constraint language is that it can only handle hard constraints. In most scheduling systems, it is recognized that there may not exist a schedule that meets all the constraints, and the system will allow some constraints to be relaxed. Most planning systems, however, have only dealt with hard constraints. There is a notion of priorities or preferences here that needs to be implemented; if a plan can't satisfy all the constraints, then the least important ones should be relaxed first.

SIPE-2 also supports a context mechanism whereby a user can explore multiple plan

options concurrently. This is a necessary capability for the CINC's staff, who are generating multiple COA's for a crisis so that the best one can be selected. The context mechanism builds hierarchical trees of alternative plans just as the plans themselves are hierarchical. When one path seems to lead to failure, the user can back the system up to a previous context and try another branch.

Last, but certainly not least, in the features of SIPE-2 that are valuable for crisis action planning, is the ability to support replanning during execution given some new information about the world. A plan that has been generated by SIPE-2 retains special nodes that contain the rationale for adding an action to the plan. There are also "phantom" nodes in a plan (usually not displayed) that are reminders of preconditions that we thought would not need to be explicitly achieved because some other part of the plan ensured that they would be true. The information in these special nodes is crucial for replanning tasks where something unexpected happens, say making a precondition false that was expected to be true during execution. SIPE-2's execution monitor can accept predicates about the state of the world, and it has several mechanisms for repairing a plan under various circumstances.

The demonstration of SOCAP in January 1992 focussed mostly on plan generation, but future work will put emphasis on the replanning problem. In crisis action situations the state of the world is changing rapidly, and no one is sure if a plan will actually succeed in its execution. A goal for a future demonstration will be to feed the output of a running simulation (predicates about conditions that have been made true or false) directly to the SOCAP planner so that SIPE-2 can repair the plan on the fly and still achieve the goals of the mission.

4 Conclusion

Despite the difficulties in applying SIPE-2 to the SOCAP domain, it was a very good choice because SIPE-2 has had as its main design goals user interactivity and efficiency. Historically, other generative planners have not been as strong in these two areas, a problem which has hindered their transition from the laboratory to the operational environment.

SOCAP has successfully demonstrated that the use of state-of-the-art AI planning technology can speed up the crisis action planning process, giving commanders more time to consider a greater number of alternative COAs before selecting one and to fully analyze an operations plan before embarking upon its execution. This is in contrast to the current situation where action sometimes must be taken in the absence of a fully developed plan.

The lasting impact of the SOCAP demonstration will be to facilitate the acceptance of generative planning technology, and hopefully artificial intelligence in general, by the military and other highly operational organizations.

Acknowledgements

This work was performed by SRI International for the DARPA/Rome Laboratory Planning Initiative under contract number F30602-91-C-0039. I wish to acknowledge Marie Bienkowski and Roberto Desimone from the SOCAP team, and David Wilkins, who developed SIPE-2, for their contributions to this paper.

References

[1] Bienkowski, Marie, "Initial Requirements Analysis for Automation in Support of Joint Military Planning", Technical Report ITAD-2062-TR-92-40, prepared for Rome Laboratory under contract number F30602-91-C-0039, March 1992.

[2] Desimone, Roberto, "Lessons Learned in Applying SIPE-2 to the Military Operations Crisis Action Planning Domain", AAAI Spring Symposium, Stanford, CA, March 27, 1992.

[3] Wilkins, David and Desimone, Roberto, "Applying an AI Planner to Military Operations Planning", Draft for publication in Morgan Kaufman volume on intelligent scheduling systems, July 24, 1992.

N93-32132

Combining Qualitative and Quantitative Spatial and Temporal Information in a Hierarchical Structure:
Approximate Reasoning for Plan Execution Monitoring

Louis J Hoebel

Rome Laboratory / C3CA
Griffiss AFB, NY 13441
hoebel@aivax.rl.af.mil

Dept. of Computer Science
University of Rochester
Rochester NY 14627-0226

Abstract

The problem of plan generation (PG) and the problem of plan execution monitoring (PEM), including updating, queries, and resource-bounded replanning, have different reasoning and representation requirements. PEM requires the integration of qualitative and quantitative information. PEM is the receiving of data about the world in which a plan or agent is executing. The problem is to quickly determine the relevance of the data, the consistency of the data with respect to the expected effects and if execution should continue. Only spatial and temporal aspects of the plan are addressed for relevance in this work. Current temporal reasoning system are deficient in computational aspects or expressiveness. This work presents a hybrid qualitative and quantitative system that is fully expressive in its assertion language while offering certain computational efficiencies. In order to proceed, methods incorporating approximate reasoning using hierarchies, notions of locality, constraint expansion and absolute parameters need be used and are shown to be useful for the anytime nature of PEM.

1 Introduction

The problem of plan generation (PG) and the problems associated with plan execution monitoring (PEM) have different temporal and also spatial characteristics in most cases. Only rarely is quantitative, metric or absolute temporal information available during PG. Most often temporal and spatial information is given as relation constraints between temporal objects and spatial objects. During plan execution the simplest and most intuitive form of information available will be metric temporal and spatial data. Some fact or event will be observed, reported or in some manner known. Monitoring the progress and continuing plausibility of a plan during its execution requires relating this metric, point information with the mostly symbolic, qualitative information that the execution is based on. A further consideration during PEM is that of bounded computational resources, which may not be present during PG.

Consider a complex system with a manager and multiple execution agent's. Further, let the agents be semi-autonomous (SA). They have some knowledge and deliberative ability enabling limited action on their own. The problem is how to monitor an executing plan for the continuing basic physical plausibility of the plan's actions and effects.

We do PEM to determine the validity or plausibility of a plan successfully executing. For negative determinations, it's important to be able to control plan execution in terms of halting execution, selecting options, replanning or exhibiting some other desired behavior. This is problematic when the plans may be denoted with qualitative symbolic representations and the update information available during execution is mostly quantitative data points. Detailed micro-monitoring and constant translation between qualitative and quantitative representations may require more bandwidth and computational resources than are available. Coarse grained monitoring may miss subtle interactions and interconnections.

The need and use of temporal modification in plan reasoning is well established [McDermott82, Allen&Koomen83, Dean85]. Alternatives to state based paradigms have considered planning as qualitative temporal constraint reasoning [Allen&Koomen83, Allen91a, 91b]. We consider a plan as a partial ordering of actions; the actions having extent in time and space. We add spatial modifications to plan reasoning arguing that this is a dual of temporal reasoning (see for example [Cui etal92]).

1.1 Goals of the Research

First we develop a scheme for PEM based on spatial and temporal hierarchies to record the "where and when" of the domain. These hierarchies are developed for partitioning of space and time (s+t) and the inclusion of metric and point data. This research aims to develop mechanisms to implement the hierarchies for general reasoning about plans. Second, we integrate qualitative and quantitative information into the s&t hierarchical structures. The partitioning of s&t hierarchies will not be perfectly efficient (see Sect. 4) and this integration will augment the qualitative relations with quantitative information This will enable representing and reasoning about disjunctive relations within the partitioning hierarchy. This representation is then used as the basis for developing an efficient computational approach to achieving approximate results for tasks such as PEM.

Exact results can be obtained for the intended dynamic and time constrained applications but this is not the expected mode of operation. The goal is a system which is able to respond in a timely and appropriate manner.

We use quantitative information on interval endpoints and durations to extend work in reasoning with hierarchical and abstract/expansion/aggregate data structures. The quantitative information is used within an approximate disjoint partitioning of the hierarchies (encoded with reference intervals) to preserve information. Quantitative point and duration values are assumed to be bounded intervals that encode the uncertainty of the extent of a domain object. An example using spatial-temporal information to reason about plans is presented. We also discuss this approach as applied to a Constraint Satisfaction Problem (CSP).

1.2 Background and Motivation

Recent work in plan generation addresses the reality that execution is no longer assumed to be in an idealized world where every action has its intended effect [Hanks90, A-I&S88]. The problem now becomes how to know what is happening or has happened and how to alter execution accordingly. Expectations of the world state are not always realized and exogenous events occur such that a sequential state transition model does not adequately reflect the complexity of the worlds in which agents will be expected to act. Implicit here is that events don't always occur when or where expected. There is some uncertainty as to the actual occurrence of an event or action. There is also uncertainty as to the spatial and temporal aspects of an event or action. We are concerned with this latter uncertainty.

Even with valid metric information, propagation through a constraint network and validation of the network may take more time than is available [Dechter et al89, Van Beek90]. Not only do we need to control the search, in both update and query, but in some applications an *anytime* response may be required [see, Dean&Boddy88]. Further, given more time a better response is possible.

One potential solution to the complexity of the problem is the use of abstraction hierarchies and reference intervals [Allen83, Koomen89] that in effect partition the search space of temporal and spatial information. The motivation here for considering space *and* time is that essentially everything, every action, every object and agent, must be somewhere at sometime. This is generally the minimum information we need to know and that is available during PEM. The combined knowledge of the two hierarchies further partitions the search space The problem is to construct appropriate hierarchies. The use of user defined reference intervals and automatically generated reference intervals [Allen83, Koomen88, 89] has been proposed but problems remain with cross connections between reference intervals creating flattened structures. Planning techniques such as abstraction [Kautz87, Tenenberg88, Hanks92] may also provide structure within applications along with certain monotonic characteristics [Teneberg91] that make them attractive for PEM.

An approach to the reasoning needed to support anytime response is the relaxation of constraints in a constraint network [Mackworth77]. Constraints are given distance bounds between nodes in a graph [Dechter et al89]. This could also be applied to a spatial (3D) network although the result is not likely to be intuitive nor computationally attractive. Multiple constraints between nodes can be relaxed to a single aggregate constraint. This transforms the network into a simple constraint network with faster computational characteristics. The network might now allow values for which no singleton solution exists. We expand the constraints into convex intervals (*convexify* in [Ladkin86]) as "aggregate constraints". Each step of this relaxation includes all plausible values given as constraints. It may now include previously excluded values from when exact methods are employed. Plausible here means that it could be a member of a valid plan or network given some combination of constraints and relations. In [Shafer&Pearl90] *plausible reasoning* is offered as "reasoning that leads to uncertain conclusions because its methods are fallible or its premises are uncertain". Implicit is that there is evidence or reason to make the conclusion.

The differing requirements of PG and PEM require not only different representations of time and space but also the unification of the differing representations and information available. The problem is to combine the qualitative PG notions with the metric data of PEM. An abstraction hierarchy is capable of incorporating information at various levels of detail in both interval (qualitative) and point/metric (quantitative) forms. The use of space and time provides further structuring of information and at a level useful to the PEM problem in a domain independent way. This structuring then is able to support both uncertainty (as to interval bounds) and anytime (albeit approximate) response in support of PEM.

2 Approach to the Problem

In a real world of time constrained situations, the larger driving question is: "what is the relation between a complex planning system and the real world". It is no longer realistic to consider a planner as only a static-world, omnipotent, plan generator [Allen&Schubert91]. We consider a planning system to consist of a deliberative or reasoning element, an execution element or agents and also a monitoring element. In short, *the planning system* needs to interact with the world. The approach we take to this interaction is to monitor the world in a flexible, adaptable way that can mesh real world observations with a projected world model. This in done in the basic parameters of the real world, i.e. space and time, and in

the constraints of executing in the real world, i.e. time-constrained.

Plan execution agents, or simply agents, that execute in the real world must be able to respond to unexpected external events in a timely and spatially coherent manner [Dennett87]. Temporal and spatial organization of knowledge can aid in monitoring and mediating an executing plan or an agent's decisions and recognition of significant events in constrained situations. The representation of temporal and spatial properties is integral to the way we are able to reason about such properties.

Specifically, we wish to look at the monitoring of an executing plan for which we have a projected world model for that plan. Reasoning about a plan's methods or goals is done by domain specific deliberative units and is not considered a part of the monitoring problem. We look at what information is true or known, when and where. The problem is that of (response) time and relevance of the new data. It is not practical to completely propagate each and every data point sensed or reported during monitoring against all known constraints. Also, data points might fit into one or more disjunctive or conditional projections (scenarios) of the executing plan. It might be more efficient to consider the union of these scenarios rather than each one separately. In short, not all data from monitoring should be assumed to be processed completely (excessive data bandwidth), nor would we wish to process all data to the same extent (data relevance).

The approach we take relies on hierarchical knowledge organization, aggregation of constraints, and integrating metric, point and absolute information into mostly qualitative and symbolic representations. We begin with a generated plan and a knowledge base of projected events, actions and facts based on the execution of that plan. This is assumed to consist of domain information that is temporally and spatially modified. The sentences in our data-base are triples of: *domain-object, space-object.* and *time-object.* The problem is now to determine when the projected course of events differs from, or can no longer be supported by, the observed state of the world. By combining qualitative and quantitative (metric and absolute) information we can monitor the progress of the plan execution vs the projection of the plan's intended actions and effects.

The temporal and spatial modifiers are organized into separate expansion hierarchies. The hierarchy is artificial in that it servers to cluster objects into reference intervals for time and space with well defined relations between the clusters. This network of qualitative constraints on the reference intervals, and the objects in the intervals, serve to partition the search space. Each object, including reference intervals, is then augmented with metric information as it becomes available. This information is used in combination with the qualitative network to enhance the overall expressiveness of the system. An aim of this research is to include metric information from plan monitoring to enable the control of propagation and search in the network of s+t relations.

2.1 Monitoring and Updating

We choose to look at plan execution for several reasons. The most obvious is that generated plans are meant to be executed. Successors to the state-based models of [Fikes&Nilsson72, Green69] include abstract plan operations and partial plans that can only be fully completed or known upon execution [Wilkins88]. In other words, monitoring is necessary to know what or how to complete and execute in the plan. Learning systems such as in [desJardins92] seek general solutions for autonomous agents in complex domains. Our approach to monitoring is not as a stimulus to behaviors but as a rational component in the interaction between a reasoning system and the real world. This includes the need to consider the uncertain and dynamic character of the real world in which plans are executed.

2.2 Expansion Hierarchies

In both PG and PEM, abstraction hierarchies can be used to reduce the search space and number of possible plan instantiations or completions [Sacerdotti74,Tenenberg88, Kautz87]. This method typically applies to actions, objects or types which have similar or common characteristics and then abstracts this commonality into a higher, generic or more encompassing classification or type. As an example we might abstract all the types of actions or series of actions that achieve some specific effect, call it X, into a pseudo-action (Abstract-achieve-X). This abstract pseudo-action is then just a symbol or place holder for one of the specific actions (specific-achieve-X1, specific-achieve-X2 etc) that actually achieve the effect X.

THE S&T HIERARCHIES

We propose a hierarchy structure that is slightly different from the abstraction presented above. The granularity of the parameters that modify a domain statement are variable within spatial and temporal hierarchies. In general, we allow for the increase or decrease of the range of domain statement modifiers. This affects the range over which these may be reasoned about or assumed.

In a hierarchy, an interval I is lower or subordinate to a reference interval I' if and only if:
1. I is a subset (included) in I' or
2. I is a subset of I'' and I' is a reference interval for I''.

One problem that can arise is that the hierarchy which is created is not structurally the one that is needed. The structure of the hierarchies should provide information. If there is excessive overlap or disjunction in the time or space constraints of domain statements, the purpose of the

hierarchy is somewhat defeated. The solution to this problem begins with the explicit addition of constraints and quantitative information on the bounds of the objects in the hierarchies and on the hierarchical structure itself. We focus on time and space as useful partitions of any KB since it is not often that we wish to know just of the existence of some object. We can then use this structure to reason about the KB in a variable and appropriate manner that may depend on the KB, the request or the specific data. It's a question of relevance.

THE AXIOMS OF S+T HIERARCHIES

We use a surface representation in the form of triples. Each form consists of a fluent, proposition or domain object *D* (the sentences of the domain) with spatial modifier *S* and temporal modifier *T* over which the domain statement is true. This is a syntactic variation of Holds(D, S, T). The interpretation is that D holds at least somewhere over S and T. Negation of the fluent or object *D* is not allowed but only negation of a triple. This has the interpretation that D does not hold anywhere over (S,T). Compared to Shoham's boundary condition K(t,[¬]p) an additional space parameter is added and negation moves outside [Shoham86]. The interpretation of negated triples (as ¬p) is the same but non-negated triples have the "plausible" semantics stated above.

$\forall$ S',S, T',T (S' $\supseteq$ S, T' $\supseteq$ T) . [D S T] $\Rightarrow$ [D S' T'] [A1]
$\forall$ S', T' [D S' T'] $\equiv \exists$ S,T (S' $\supseteq$ S, T' $\supseteq$ T) . [D S T] [A2]

As a logical consequence of A2 the following theorem for negated triples is stated.
$\neg$[D S' T'] $\Rightarrow$ $\forall$ S, T (S' $\supseteq$ S, T' $\supseteq$ T). $\neg$[D S T] [T1]

The first axiom states that if a domain sentence is true somewhere in (S, T) then it is true in all expansions that include (S,T). Conversely, if we know that a domain sentence holds over (S,T) then we know of the existence of some refinement of (S,T) where the sentences holds. Theorem T1 states the implication of a sentence that does not hold anywhere within (S,T). Negated triples mean that the domain sentence is not true in all refinements of (S,T). There is similar to standard Kripke possible world semantics. Also, *facts, events, properties* and other types of domain object types are not introduced. This presents triples as unitary notions of spatial and temporal propositions and follows [Shoham86a] in that regard.

3 S&T Examples

The following example shows the monitoring and reasoning about agents using a spatial expansion hierarchy and temporal information with metric data. System operation is introduced first, then a simple example showing the usage of temporal and spatial hierarchy information in execution monitoring. Next an an execution with an indeterminate situation is presented along with how a simple spatial constraint is used to resolve a potential conflict. This resolution provides a simpler, less complex constraint than temporal constraints alone would require.

3.1 System Operation

The execution monitoring system provides a mechanism for interpreting the status of an executing plan in terms of s&t consistency. Its provides a means of integrating metric and point data and also vague (in terms of s&t parameters) information into various types of plans. This includes qualitative and quantitatively constrained, partially ordered and abstract plans.

Given an update (sensor report etc) of the form [D s t], where s and t are points, is there a [D S T] s.t. s $\in$ S and t $\in$ T and [D S T] is plausible in both the real world and the projection of the executing plan. Also, given an update of the form [D S' T'] where D holds of at least some of S' and T', is there a [D S T] s.t S' $\supseteq$ S and T' $\supseteq$ T and [D S T] holds?

Updates relative to the s&t hierarchy are made for each triple that is consistent. The bounds on the start time and finish time of the appropriate interval T are modified as appropriate and also the bounds on the spatial modifier S. This is done in conjunction with what duration and scope information is known about S and T respectively.

For example (with t being an initial report of a event):
[T.start < t] & [T.finish > t]
[t + dur.max > T.finish] & [t - dur.max < T.start]

For T in T' (a reference interval), updates to T' reflect the constraints imposed by T, the relation between T and T' and metric information. Also any interval relations within T' are updated using an Allen style interval reasoner and the propagation of metric constraints occurs [Kautz&Ladkin91]. If there is no change to T' and T has no overlapping relations or metric constraints with intervals outside of T' then the update procedure is finished. If T' is updated (bounds change) or T has overlapping relations out side of T', then the updating proceeds on T' similar to that of T but at the next level in the hierarchy. The update of metric relations outside of T' also needs to be checked for consistency.

3.2 A Transportation Domain

In this example, we start with a symbolic, abstract plan in a simple transportation domain. The domain consists of two cities connected by segments of train tracks which make-up various paths between the cities (see [3.2]). Traveling over these segments of track are two train engines that operate as SA agents. We assume the agents have a map of the domain. An agent's task in this domain will consist of traversing a path from one city to another. We will monitor the agent's actions and the spatial-temporal aspects of plan execution. This scenario is

complex enough yet general enough to demonstrate our approach while using a constrained space of the track segments that simplifies some details for now.

Consider the following plans for the agents to execute: Agent *Eng1's* task is to travel, via some route denoted *Path1*, from city A to City B. The time during which this plan is to be executed is called interval *T1*. A route here is a non-cyclic sequence of track segments. The primitive actions that we consider are at the level of segment traversal by the domain agents. Similarly, *Eng2* is to travel along a route *Path2* from city B to city A during a time interval *T2*. These agents are autonomous in path selection in that no constraints have been given for path selection. Likewise, the time intervals are initially unconstrained by the planner, although as we shall see the tasks and topography constrain the plans as execution proceeds.

The agents are making the routing decisions and may initially determine complete routes or postpone decisions until sometime during execution, perhaps relying on local conditions. *Eng1* begins with an abstract plan to reach the destination city over the unspecified route *Path1* during time *T1*. Let the plan or situation be represented to the plan monitoring component (PMC) as a triple: [Eng1 Path1 T1]. For the spatial-temporal monitoring component of the system this represents the fact that *Eng1* is at or in location *Path1* during time *T1*.

Since the agent must act on these abstract plans, concrete actions result. A representation of the plans specified to the agents is received by the PMC of the system. The PMC receives updates of the agents activities which consist of time and location information. The PMC must be able to monitor these activities and integrate this information with the overall plan. Further, we want to be able to determine if the information received by the PMC is consistent with its view of the world, i.e. its model, plans and projections. Additionally, unanticipated actions may be occurring and also need to be monitored for interaction with the executing plans. Only relevant information should then be directed to an agent. We are not considering agent-to-agent communication for now, nor other complex agent-agent interactions such as negotiations, recognition or acknowledgement as these are outside the scope of this work.

Consider now the high level abstract representation of the situation described above. In our projected knowledge-base there are the representations of *Eng1* and *Eng2's* plan actions. The essential part for monitoring is the movement during the two time intervals and over the space S of the domain (see [3.2]). S represents all space in the domain and similarly there is a root time interval T that contains all other time intervals of interest. These are the most abstract or expansive values in their respective hierarchies.

Initial KB: [Eng1 Path1 T1] [Eng2 Path2 T2] [3.1]

T1 and *T2* are unconstrained time objects; essentially time intervals with additional metric information attached. The symbols *Path1* and *Path2* both denote the most general or least constrained spatial value: S. S is the expansion or generalization of all possible paths within the domain. We can represent any path (see [4.2] below) thru the domain as the disjunction of all segments making a path from CityA to CityB. In a more dynamic and complex domain it is reasonable to think that agents begin with abstract plans and leave the exact instantiations to be done later as necessary. The planning system later constrains the agent's paths based on local information or changing situations. Here we are calling spatial information "paths" when a more general terminology might be location or spatial scope of a domain object, predicate, event or action.

The AND/OR representation of all the paths is a relational constraint description of the spatially simple domain paths that connect CityA and CityB. This spatial domain is restricted to the segments and their connections. The explicit qualitative constraints between segments are limited to *equal, meets, met-by, contains* and *disjoint*. We see these are a subset of Allen's temporal interval relations [Allen83]. The relation *contains* is added for constraining the expansions in the spatial hierarchy. An example of this is: S1/S2 *contains* S1. This reinforces our intuition that temporal and spatial reasoning are similar and can be used and supported with similar techniques. More complex domains will make use of additional spatial relations (*above, with-in, below, north-of, etc*) as necessary. A spatial object in the domain is a segment or a sequence of path segments. This is shown below with implied ANDs while choice points are indicated explicitly by ORs.

```
[S => (OR (S1 S2)                                    [3.2]
          (S3 (OR (S4)
                  (S5 S6)) S7) ]
```

In the example given here we have a spatial expansion hierarchy for this domain that partitions the space *S*. Each expansion space is disjoint from each other at the same level. Each node within the space is labeled with the track segment or path abstraction represented by that node. Each non-leaf node is an abstraction or expansion of the nodes below it in the tree. The leaves of the tree indicate the possible exact instantiations of the space above it.

3.2.1 Example with no conflict

In this example, we start with the two abstract plan representations in [3.3] and the static spatial data of [3.2]. Each agent knows its own plans and the spatial information about the domain. The PMC of the planning system has this information plus receiving reports during execution time. From this situation the PMC needs to

recognize conflicts and the planning system needs to respond with additional constraints on the agents to ensure conflict free path selection.

At time 0900, agent Eng1 reports leaving city A and Eng2 reports leaves city B and the current track they are located on.

Added to KB: [Eng1 S1 0900] and [Eng2 S7 0900] [3.3]

These data require *Path1*, of [3.1], to have the value *S1/S2* at the most abstract level and *Path2* to point to the value *S3/S7*. The segments S1 and S7 are in completely disjoint subspaces of the S hierarchy. No spatial conflict is anticipated between *Eng1* and *Eng2* during the time intervals *T1* and *T2* and any further analysis will yield no additional information. If resources become constrained, the PMC could focus its monitoring resources to situations in which conflict is anticipated.

We see here the need to be able to incorporate metric information into the temporal representation. Our time objects are temporal intervals, symbolically represented but with additional quantitative information about end points and durations. Hence calling them objects rather than simply intervals. We have the absolute time values of [3.3] to incorporate into the initial interval representation. This is also true of the spatial representation as a numeric "mile post" could have been given instead of a track segment name. What is required is a method of determining the correct location in the hierarchy for the given metric information. Here *T1* and *T2* record a latest start time of 0900 for the intervals *T1* and *T2*. If durational information about the transit times of the segments is known then we could generate bounds on the end points of the time intervals. This would be useful in limiting unnecessary constraint checking. What should be noted here is that the reports coincide with the plan and that this argues for minimal action. It's what is expected. Any additional report that places *Eng1* on *S1/S2* and or *Eng2* on *S3/S7* does not not force further checking of constraints.

3.2.2 Example with conflict

Starting with the same abstract plans and spatial information, let us consider another situation. The domain constraints precluded more than one agent to traverse a segment at any one time. Let the reports indicate that some conflict is possible but not inevitable. Each agent starts out on paths that may, when fully specified, be in some conflict.

[Eng1 S3 0900] and [Eng2 S7 0900] [3.4]

Given this as the initial KB, both *Path1* and *Path2* now point to the same node, namely *S3/S7*. There exists both conflict and non-conflict routes which might be taken given this initial report. The necessary information is added based on the spatial decomposition of *S3/S7* and generate appropriate time intervals.

[Eng1 S3 T11] [Eng2 S7 T13] [3.5]
[Eng2 S7 T21] [Eng2 S3 T23]

Where T11 and T13 are contained during T1 and likewise for T21, T23 and T2

There is no simple, single temporal constraint that will ensure success, given what we know. By generating temporal intervals for each agent on both *S3* and *S7* and then constraining the appropriate intervals to be disjoint, we can avoid contention for *S3* and *S7*. These are the known segments that must be traversed. The problem is that the conflict, using temporal considerations, most likely occurs on segments *S4* or *S5* and *S6*. Using temporal constraints alone would over constrain the problem by generating disjoint time intervals for each agents actions while in *S4/S5/S6* space. This is unnecessary depending on the actual path taken in this region and we wish to avoid generating new and perhaps unnecessary constraints. Additionally the possibility of contention for *S3* or *S7* might be heightened by possibly delaying an agent from entering the *S4/S5/S6* region due to a disjoint constraint temporal constraint. Here, a simple spatial constraint is available and more desirable.

The simplest constraint is then a spatial constraint on each agent's path selection. Under the *S3/S7* node in the S hierarchy we constrain the agents selection at the first OR node choices. This way we make the minimum number of constraints necessary.

[Eng1 S4 T12] and [Eng2 S5/S6 T22] [3.6]

As noted before, *T11* and *T23* are disjoint intervals and the same for intervals *T21* and *T13*. If durational information about the transit times of the segments is known then we might not need to post the disjoint constraints and also do the subsequent consistency check. We would be able to determine that the intervals are disjoint based on statistical information about transit duration. Details are available in [Hoebel92].

We see from these examples that spatial and temporal constraints can be useful in the proper interpretation of reported data. This compound approach can also lead to simpler constraints than temporal reasoning alone. Monitoring leads to intervention (or not) at an appropriate time and in an appropriate manner.

4 Current and Proposed Work

Partitioning of the search space is one way to reduce the complexity of search and hence the time needed during query or update [Dean86]. In this section we employ techniques consistent with hierarchical knowledge

structure and show how metric information extends the restricted qualitative model.

4.1 Extending a Restricted Qualitative Model

In [Kahn&Gorry77, Allen83 and Koomen89 a reference hierarchy is proposed with the main motivation being to reduce the space requirements. Koomen differs in that he rejects the notion of a predefined reference hierarchy and also the notion that structure should be left to a higher level reasoning system. He describes a system where a "reasoning system itself could structure the network dynamically on the basis of posted and derived constraints, and do it without losing information"

One approach to an efficient implementation is to have a structure based on interval relations in the s+t hierarchies that are non-overlapping. Koomen refers to this as a "containment-based" reference hierarchy. We extend this interval based model in two ways. First developing procedures for overlapping relations to exist between intervals of non-containing (disjoint) reference intervals without flattening the hierarchies. Implicit in this is disjunctive constraints. Second we add metric, point and durational information which enables the management of the reference hierarchies while limiting propagation. This has attractive space requirements while providing computational mechanisms in moving beyond interval and relative duration information.

This extension now allows all binary (point and interval) relations to be represented while avoiding flattening the hierarchy. A full reasoner, interval, point, metric or combined, applied within reference intervals of a reasonable size is still very efficient [Koomen88, 89 Kautz&Ladkin91]. A full reasoner for spatial relations would be obviously more complex but it is reasonable to believe that a subset of spatial interval relations can be developed that is both adequately expressive and efficient [Cui etal92] . The problem is how to obtain and manage an appropriate s+t hierarchy. In the dynamic environment of PEM, hierarchies necessarily must be flexible to the changes and uncertainty that occur in the world. The semantics of the s+t expansion hierarchies provide the flexibility via the granularity of the s+t parameters. The separate handling of disjoint/contains and overlaps constraint relations is more fully adequate in terms of computation, expressiveness, the incorporation of metric point data and for exact or approximate reasoning than previously proposed systems.

4.2 Adding Metric Information: control of propagation and search

The inclusion of global constraints or chains of *overlapping* relations can lead to a rapid flattening of a reference hierarchy [Koomen89]. The problem generated by these overlapping interval relations can be over come by keeping the disjoint/contains reference hierarchy intact while explicitly and exactly encoding the overlapping relations which exist between domain intervals and reference intervals. This method allows a "meta-graph" of reference intervals for faster search while restricting propagation and consideration of the overlap relations only when appropriate. The reference hierarchy can be efficiently managed with the addition of metric and point information. Using this information, a disjoint/contains hierarchy can be created and managed even when such a structure does not exactly exist.

RELATIONS WITHIN A REFERENCE INTERVAL

Interval relations within a particular reference interval are intended to be reasoned about completely. A full reasoner for this task would use qualitative constraints as in a Allen style interval reasoner plus a point algebra and metric information. A system such as MATS appears to be able to handle this task in low order polynomial time and as such is considered adequate for the task. A table of qualitative interval relations is maintained while metric and point information is stored with each interval object. For intervals entirely within a single reference interval, all updates and queries concerning these intervals are assumed to be handled by the full reasoner. Each interval also maintains explicit overlaps information for overlapping relations outside the reference interval. This is for asserted information and does not imply that deduced relations are stored.

OVERLAPPING BETWEEN REFERENCE INTERVALS

Particularly during PEM, hierarchies need to be flexible and dynamic in structure. This is true even for those with an initial disjoint/contains structure. For example, updates to intervals within a reference interval may extend an interval into an overlapping relation. Consider the simple case of an action, event or goal of an executing plan having a longer duration than was initially conceived of in the plan. Updating the bounds of such an interval and its parent reference interval may now create overlapping conditions with other intervals. Rather than collapse the reference intervals of these now overlapping intervals into a single and larger reference interval, the overlapping relation is noted and reasoned about as an adjunct activity during search and propagation. Point and metric information can be used to determine the extend of the possible overlap. This information can prevent unnecessary computation by limiting search and propagation to the relevant areas of the hierarchy. Metric, point and crucially duration information allows the system to quantify the extent of overlap and respond accordingly.

UPDATING CONSTRAINTS, HANDLING ASSERTIONS {UNARY, BINARY, METRIC, QUALITATIVE}

New assertions or updates may require the interval or intervals concerned to be updated with new information. Any changes are first made to the interval objects

themselves. Changes then propagate downward as the updated interval may also serve as reference interval. A full reasoner is then applied to the entire reference interval containing the updated interval. After the full reasoner propagates constraints throughout the reference interval, affected intervals that serve as reference intervals propagate these changes downward. Finally, reference interval changes are propagated upward through the containing reference hierarchy and outward to sibling reference intervals. Note that procedures are called only when changes are made to intervals. No procedure is called on intervals outside the reference interval unless a change occurs that affects the interval. Also note that any assertion or propagation that effects an overlapping inter-reference interval relation is handled as a separate subprocedure during the propagation procedure that effects the overlapping intervals.

SUMMARY OF ASSERTION UPDATING STEPS

1. Update intervals (endpoint constraints, durations and explicit relations)
2. Propagate interval update downwards (recursively)
3. Apply full reasoner to all members of the intervals reference interval (those that change are queued)
   3.1 Update to overlapping reference intervals outside the initial update interval's reference interval. (keeping a queue of intervals that are changed)
4. Propagate changes to reference interval (and queue reference interval if changed)
5. Select next interval in queue.

In the case where we know that reference intervals may overlap but not know the extent of the overlap, we are faced with a choice. One method is a full reasoner used over the combined reference intervals to detect all possible relations. Although this is possible, it is not advisable in general since this is the exact problem of hierarchy flattening that we are trying to avoid. The simplest course is to do nothing and wait until information is obtained that reveals the extent of the overlap. A more balanced approach is to require some estimate of the duration of each domain object, activity, fluent, etc. This of course is only a middle ground between knowing nothing of the extent and of knowing exactly the extent. As the system reasons over longer spans in the hierarchy, the more the durational uncertainty is going to accumulate and tend towards ignorance of actual overlap extent.[1] In practice we expect the structure of the problem to be such that propagation is generally limited to local areas of the hierarchies. Overlapping relations will not tend to propagate endlessly. The CSP example in the next section shows how absolute times, even when dealing with expanded constraints, limit propagation quickly and provide accurate bounds.

[1] A separate issue to address is the characterization of this durational uncertainty.

USING METRIC CONSTRAINTS

When metric information, point information and duration information is added to symbolic interval constraints, the ability to determine exact relationships is increased. Consider for example three intervals A, B and C and the constraints *A overlaps B* and *B overlaps C*. A and C are related by any of the relations *before*, *meets*, or *overlaps*. We can determine if A does overlap C with the addition of metric and endpoint information for each interval. This is something that is not possible with strictly qualitative reasoning over intervals. Given bounds on the start, finish and duration times of the intervals, the determination of possible relations can be made. These bounds may or may not be the same as a "stated constraint" on the endpoints or the duration of an interval. We may be able to infer a tighter constraint. This is shown in the example in sect. 4.3.1. For this reason it may be desirable to track both stated bounds and an inferred (non monotonic) bounds.

Lets continue the case of intervals A, B and C as described above. With additional metric information about the start and finish times we may be able to determine a more exact ordering of the endpoints. We now impose the following three metric point constraints:

[A.finish - B.start] < 30
[B.finish - C.start] < 30
[B.finish - B.start] > 60

The third constraint requires a minimum duration for interval B that requires A before C. This type of information is useful in the determination and management of of intervals in reference hierarchies. Consider now the case in which slightly different constraints allow for a finite amount of possible overlap. Consider the situation, in the notation of Allen, (A :o :m :b B :o :m :b C):

[A.finish - B.start] < 35
[B.finish - C.start] < 35
[B.finish - B.start] > 60

These constraints now allow for A and C to overlap but only by up to 10 units. This does not require that A and C be contained within a single reference interval. We can place A and C in disjoint reference intervals, note the overlap and then use the explicit information when required. Here we can note the constraint between B and both A.finish and C.start. This information would propagate to the reference interval containing A and C. Subsequent updates to the reference intervals need not propagate across this "link" unless the "link" itself (the endpoints) is in some manner involved (changed).

Interval B, which might overlap both A and C, might be placed in either reference interval depending on some domain specific criteria such as causal connection or number and type of relations to the intervals in the containing reference intervals. In this way we can maintain the expansion disjoint/contains hierarchies but still deal efficiently with overlapping relations.

Additionally we can structure the hierarchies on domain information when such information is known and appropriate. This would appear to be an improvement over a predefined reference hierarchy.

4.3 A CSP model

In [Dechter et al89] an instance of the Constraint Satisfaction Problem is presented as a temporal reasoning problem. They present an approach that incorporates qualitative, metric and absolute time in a network of temporal constraints. The absolute times are stored as special nodes, nodes are temporal variables in general, and the constraints are added as arcs between variable nodes. In a longer version of this paper a transformation of the CSP is formulated that is consistent with the notions developed in the s+t hierarchies presented in this paper. First, a network is given for the example as presented in [Dechter et al89] and then a network where absolute time/arc constraints are reformulated as constraints (bounds) on node values. The special absolute time nodes are eliminated and the absolute times internalized at the variable nodes. This is done to avoid making a fully connected graph of arc constraints. Propagation of absolute times can be interrupted and restarted as they serve as something of a local constraint caching mechanism. The metric information itself can be used to control propagation and to determine currently plausible values for a node. An small extended example shows propagation of values to be localized and efficient within the hierarchy. The final model may return a weaker constraint interval than if exact methods are used. The absolute times are used to bound this weakness. A example is shown that transfroms multiple arc constraints in a Temporal CSP (TCSP) into single arc constraints representative of the Simple TCSP. The results of this relaxed form of TCSP for approximate reasoning si compared to exact singleton solutions.

We have presented the outline of a representation that allows for approximate reasoning and can clearly incorporate symbolic, metric and approximate or uncertain information. We do this in a uniform way with both space and time and thus allow for easy application to scheduling and routing problems, which is to say that it applies to plans that have all 4 dimensions as part of their solution.

4.4 Summary

Limiting the expressiveness of the qualitative model limits the search required for queries and updates of the (basically) qualitative model. We start with this model and defined two types of intervals and give the constraint relations allowed between them. An extended model is presented that seeks to over come the limited expressiveness of the first model. It uses absolute and metric time, durations, and endpoint relations to achieve control of propagation and capture the full expressiveness of relations while retaining the graph search information of the restricted disjoint/contains model and the efficiency of metric comparisons.

Finally we discussed the general techniques of an expansion hierarchy as an approximate solution to a metric constraint satisfaction problem. The goal is to produce a more fully expressive spatial-temporal model that is efficient in general and has anytime characteristics. It appears that the efficiency needed for real-time response is in conflict with the complexity of a more expressive temporal reasoner [Vilain&Kautz86, others] but can be overcome by exploiting structure and with approximate methods.

I gratefully acknowledge the Rochester Temporal Reasoning group and Leo Hartman for what sanity is presented here. All errors are clearly my own.

References:

[Allen 83] James F. Allen *Maintaining Knowledge about Temporal Intervals.* Communications of the ACM26 (no. 11): 832-843

[Allen&Koomen 83] Allen J.F. and J. A. Koomen *Planning Using a Temporal World Model"* Proceedings of the 8th IJCAI pps 741-747. Karlsruhe, W.Germany August 1983

[Allen 91a] James F Allen, Planning as Temporal Reasoning , *2nd Conference on Knowledge Representation* [KR91] April 1991, Boston Mass.

[Allen 91b] Allen, James; Kautz, Henry; Pelavin, Rich and Josh Tenenberg "Reasoning about Plans" Morgan Kaufman Publishers 1991

[Allen&Schubert 91] Allen, James. and Len Schubert; " The Trains Project", Univ. of Rochester, Dept. of Comp. Sci., TRAINS Technical Note 91-1 May 1991

[Ambrose-Ingersal&Steele88] in "Readings in Planning" Eds. Allen, Tate and Hendler; Morgan Kaufman 1991

[Cui etal92] Cui, Z; Cohn, A & D. Randell "Qualitative Simulation Based on a Logical Formalism of Space and Time" Proceedings AAAI-92, San Jose CA. 1992

[Dean86] Thomas Dean, "Intractability and Time Dependent Planning" The1986 Workshop on Reasoning about Actions and Plans; Morgan Kaufman Publishers.

[Dean&Boddy 88] Thomas Dean and Mark Boddy *"An Analysis of Time-Dependent Planning"* Proceedings of the 7th AAAI, Aug. 1988 ppg 49-54

[Dean&McDermott87] Dean T.L. and McDermott D.V. Temporal Date Base Management *AI Journal* 1-55 1987

[Dechter et al89] Dechter, R., Meiri, I., and Pearl J. *Temporal Constraint Netwroks* Proceedings of the Conference on Knowledge Representation [KR89] pps 83-93 Toronto, Canada May 1989

[desJardins92] Marie desJardins, "PAGODA: A Model fro Autonomous Learning in Probabilistic Domains" Univ. of Calif at Berkeley, UCB/CSD 92/678 1992

[Dennett87] D. Dennett "Cognitive Wheels: The Frame Problem in AI" in *The Robot's Dilemma* Ed. by Z. Plylshyn Ablex Publishing, Norwood NJ 1987

[Green69] Green, Cordell *"Application of Theorem Proving to Problem Solving"* Readings in Planning, Eds. J. Allen, J. Hendler and A. Tate pgs 67-87 Morgan Kaufman Publishers 1991.

[Hanks 90] Steve Hanks "Practical Temporal Projection" Proceedings of the 8th AAAI, Boston MA August 1990.

[Hoebel92] Louis Hoebel, Tech. Report (forthcoming)

[Kahn&Gorry77] Kahn, K. and G. Gorry "Mechanizing Temporal Knowledge" AI Journal 87-109, 1977

[Kautz 87] Henry A. Kautz *"A Formal Theory of Plan Recognition"* Tech. Rep. TR-215 University of Rochester, Dept. of Computer Science, May 1987

[Koomen88] Koomen J.A. *The TIMELOGIC Temporal Reasoning System*, Tech. Rep TR-231 University of Rochester, Dept. of Computer Science, October 1988

[Koomen88a] Koomen J.A. *"Reasoning about Recurrance"* Phd Thesis University of Rochester, Dept. of Computer Science, October 1988

[Koomen89] Koomen J.A. *"Localizing Temporal Constraint Propagation* Proceedings of the 1st Conference on Knowledge Representation Toronto Canada 1989 [KR89] 198-202

[Ladkin86] Peter Ladkin "Primitives and Units for Time Specification" Proceedings of the 5th AAAI, pps 354-359, Philidelphia, PA. August 1986

[Mackworth 77] Allan K Mackworth "Consistency in Networks of Relations" AI Journal pps 99 -108,1977,

[Malik &Binford83] Malik and Binford *Reasoning about space and time.* [IJCAI 83]

[McDermott82] Drew McDermott, *A Temporal Logic for Reasoning about Actions and Plans"* Cognitive Science 6(2), 1982 pps 71-109.

[Sacerdoti74] Earl Sacerdoti, *"Planning in a Hierarchy of Abstraction Spaces"* Artificial Intelligence 115-135, 1974

[Shafer76] Glen Shafer, "A Mathematical Theory of Evidence" Princeton Press, 1976

[Shoham86] Yoav Shoham, "Chronological Ignorance" Proceedings of the 5th AAAI-86 pps 389-393, Philidelphia, PA. August 1986

[Shoham86a] Yoav Shoham, "Temporal Logics in AI" Artificial Intelligence Vol. 33, No. 1 Septemper 1987

[Tenenberg88] Josh Tenenberg "Abstraction in Planning" Phd Thesis, University of Rochester Dept. of Computer Science TR-250; May 1988

[Teneberg91] [Knoblock, C; Qiang, Y. and J.Tenenberg *"Characterizing Abstraction Hierarchies for Planning"* AAAI-91. pps 692-697 Anahiem CA, July 1991

[Traum and Allen 91] Traum, David R. and James F. Allen, "Causative Forces in Multiagent Planning" in *Decentralized AI 2* Editors Y. Demazeau and J. Muller, .Science Publishers B. V.

[Vilain&Kautz86] Vilain, M. and H. Kautz, "Constraint Propagation Algorithms for Temporal Reasoning" AAAI-86 pps 377-382, Philidelphia, PA. August 1986

[Schubert&Miller90] Schubert L. and Miller S., *Time Revisited* Computational Intelligence 1990

[Van Beek 90] Peter Van Beek, Reasoning about qualitative temporal information. Proceedings of the 8th AAAI; Boston, Mass. August 1990.

[Wilkins88] David E. Wilkins *"Practical Planning"* Morgan Kaufman Publishers, San Mateo Calif. 1988

Session A3: PLANNING AND SCHEDULING II

Session Chair: Andrew Mayer

The MICRO-BOSS Scheduling System: Current Status and Future Efforts

Norman M. Sadeh

Center for Integrated Manufacturing Decision Systems
The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 - U.S.A.

sadeh@ri.cmu.edu

1. INTRODUCTION

[1]Over the past few years, several approaches to scheduling have been proposed that attempt to reduce tardiness and inventory costs by *opportunistically* (i.e. dynamically) combining a resource-centered perspective to schedule bottleneck resources, and a job-centered perspective to schedule non-bottleneck operations on a job by job basis. Rather than relying on their initial bottleneck analysis, these schedulers reexamine the problem each time a resource or a job has been scheduled. This enables them to detect the emergence of new bottlenecks during the construction of the schedule. This ability has been termed *opportunistic scheduling* [3]. Nevertheless, the opportunism in these systems remained limited, as they required scheduling large resource-subproblems or large job-subproblems before allowing for a change in the scheduling perspective (i.e. before permitting a revision in the current scheduling strategy). For this reason, we actually refer to these approaches as *macro-opportunistic* techniques.

In reality, bottlenecks do not necessarily span over the entire scheduling horizon. Moreover they tend to shift before being entirely scheduled. A scheduler that can only schedule large resource subproblems will not be able to take advantage of these considerations. Often it will overconstrain its set of alternatives before having worked on the subproblems that will most critically determine the quality of the entire schedule. This in turn will often result in poorer solutions. A more flexible approach would allow to quit scheduling a resource as soon as another resource is identified as being more constraining[2]. In fact, in the presence of multiple bottlenecks, one can imagine a technique that constantly shifts attention from one bottleneck to another rather than focusing on the optimization of a single bottleneck at the expense of others. Therefore, it seems desirable to investigate a more flexible approach to scheduling, or a *micro-opportunistic* approach, in which the evolution of bottlenecks is continuously monitored during the construction of the schedule, and the problem solving effort constantly redirected towards the most serious bottleneck. In its simplest form, this micro-opportunistic approach results in an *operation-centered* view of scheduling, in which each operation is considered an independent decision point and can be scheduled without requiring that other operations using the same resource or belonging to the same job be scheduled at the same time.

Section 2 describes a micro-opportunistic factory scheduler called **MICRO-BOSS** (Micro-Bottleneck Scheduling System). Section 3 describes an empirical study that compares

[1]This research was supported, in part, by the Defense Advanced Research Projects Agency under contract #F30602-88-C-0001, and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

[2] [1] describes an alternative approach in which resources can be resequenced to adjust for resource schedules built further down the road. This approach has been very successful at minimizing makespan. Attempts to generalize the procedure to account for due dates seem to have been less successful so far [6].

MICRO-BOSS against a macro-opportunistic scheduler that dynamically combines both a resource-centered perspective and a job-centered perspective. A summary is provided in Section 4, along with a brief discussion of current research efforts.

2. A MICRO-OPPORTUNISTIC APPROACH

In the micro-opportunistic approach implemented in MICRO-BOSS, each operation is considered an independent decision point. Any operation can be scheduled at any time, if deemed appropriate by the scheduler. There is no obligation to simultaneously schedule other operations upstream or downstream within the same job, nor is there any obligation to schedule other operations competing for the same resource.

MICRO-BOSS proceeds by iteratively selecting an operation to be scheduled and a reservation (i.e. start time) to be assigned to that operation. Every time an operation is scheduled, a new *search state* is created, where new constraints are added to account for the reservation assigned to that operation. A so-called consistency enforcing procedure is applied to that state, that updates the set of remaining possible reservations of each unscheduled operation. If an unscheduled operation is found to have no possible reservations left, a *deadend state* has been reached: the system needs to *backtrack* (i.e. it needs to undo some earlier reservation assignments in order to be able to complete the schedule). If the search state does not appear to be a deadend, the scheduler moves on and looks for a new operation to schedule and a reservation to assign to that operation.

In MICRO-BOSS, search efficiency is maintained at a high level by interleaving search with the application of consistency enforcing techniques and a set of look-ahead techniques that help decide which operation to schedule next (so-called *operation ordering heuristic*) and which reservation to assign to that operation (so-called reservation ordering heuristic).

1. **Consistency Enforcing (or Consistency Checking):** Consistency enforcing techniques prune the search space by inferring new constraints resulting from earlier reservation assignments [2, 5].

2. **Look-ahead Analysis**: A two-step look-ahead procedure is applied in each search state, which first optimizes reservation assignments within each job, and then, for each resource, computes contention between jobs over time. Resource/time intervals where job contention is the highest help identify the critical operation to be scheduled next (operation ordering heuristic). Reservations for that operation are then ranked according to their ability to minimize the costs incurred by the conflicting jobs (reservation ordering heuristic). By constantly redirecting its effort towards the most serious conflicts, the scheduler is able to build schedules that are closer to the global optimum. Simultaneously, because the scheduling strategy is aimed at reducing job contention as fast as possible, chances of backtracking tend to subside pretty fast too.

The so-called opportunism in MICRO-BOSS results from its ability to constantly *revise its search strategy and redirect its effort towards the scheduling of the operation that appears to be the most critical in the current search state*. This degree of opportunism differs from that displayed by other approaches where the scheduling entity is an entire resource or an entire job [3], i.e. where an entire resource or an entire job needs to be scheduled before the scheduler is allowed to revise its current scheduling strategy.

3. PERFORMANCE EVALUATION

MICRO-BOSS was compared against a variety of scheduling techniques, including popular combinations of priority dispatch rules and release policies suggested in the Operations Management literature [5].

This section outlines a study comparing MICRO-BOSS against a macro-opportunistic scheduler that dynamically combined both a resource-centered perspective and a job-centered perspective, like in the OPIS scheduling system [3]. However, while OPIS relies on a set of repair heuristics to recover from inconsistencies [4], the macro-opportunistic scheduler of this study was built to use the same consistency enforcing techniques and the same backtracking scheme as MICRO-BOSS[3]. The macro-opportunistic scheduler also used the same demand profiles as MICRO-BOSS. When average demand for the most critical resource/time interval was above some threshold level (a parameter of the system that was empirically adjusted), the macro-opportunistic scheduler focused on scheduling the operations requiring that resource/time interval, otherwise it used a job-centered perspective to identify a critical job and schedule some or all the operations in that job. Each time a resource/time interval or a portion of a job was scheduled, new demand profiles were computed to decide which scheduling perspective to use next. Additional details on the implementation of the macro-opportunistic scheduler can be found in [5].

In order to compare the two schedulers, a set of 80 scheduling problems was randomly generated to cover a wide variety of scheduling conditions: tight/loose average due dates, narrow/wide due date ranges, one or two bottleneck machines. Each problem involved 20 jobs and 5 resources for a total of 100 operations (see [5] for further details).

[3]An alternative would have been to implement a variation of MICRO-BOSS using the same repair heuristics as OPIS. Besides being quite time-consuming to implement, such a comparison would have been affected by the quality of the specific repair heuristics currently implemented in the OPIS scheduler.

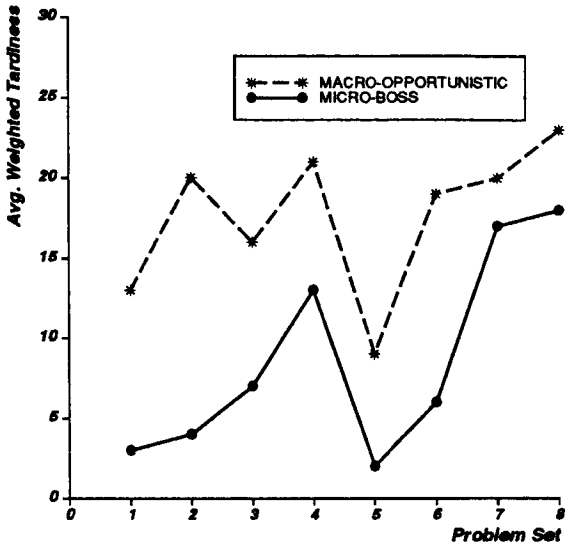


Figure 3-1: Tardiness performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

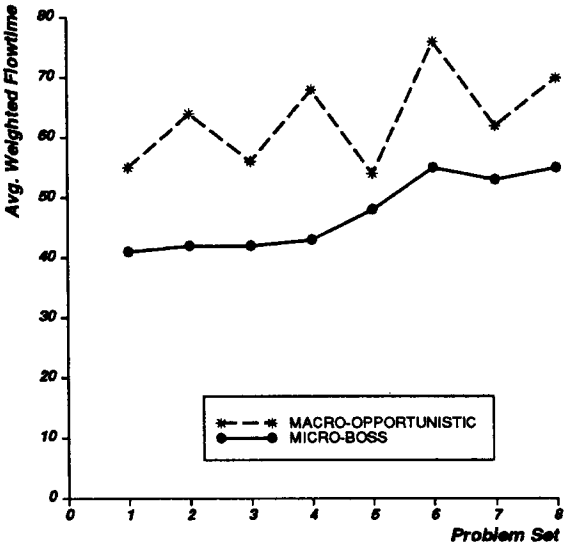


Figure 3-2: Flowtime performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

Figures 3-1, 3-2 and 3-3 summarize the results of the comparison between MICRO-BOSS and the macro-opportunistic scheduler[4]. The macro-opportunistic scheduler was consistently outperformed by MICRO-BOSS (under all eight scheduling conditions) both with

[4]The results presented in this section correspond to the 69 experiments (out of 80) that were each solved in less than 1,000 search states by the macro-opportunistic scheduler.

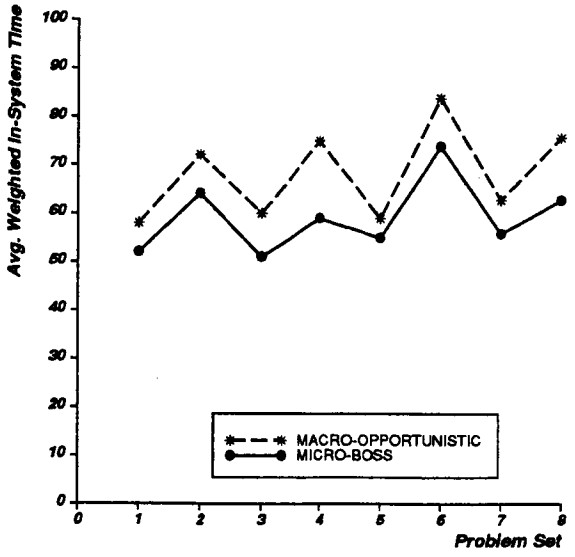


Figure 3-3: In-system time performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

respect to tardiness, flowtime (i.e. work-in-process) and in-system time (i.e. total inventory, including finished-goods inventory). *More generally, these results indicate that highly contended resource/time intervals can be very dynamic, and that it is critical to constantly follow their evolution in order to produce quality schedules.*

In most problems, MICRO-BOSS achieved a search efficiency of 100% (computed as the ratio of the number of operations to be scheduled over the number of search states that were visited), and required about 10 minutes of CPU time to schedule each problem. The current system is written in Knowledge Craft, a frame-based representation language built on top of Common Lisp, and runs on a DECstation 5000.

4. CONCLUSIONS

In this paper, a micro-opportunistic approach to factory scheduling was described that closely monitors the evolution of bottlenecks during the construction of the schedule, and continuously redirects search towards the bottleneck that appears to be most critical. This approach differs from earlier opportunistic approaches, such as the one described in [3], as it does not require scheduling large resource subproblems or large job subproblems before revising the current scheduling strategy. This micro-opportunistic approach has been implemented in the context of the MICRO-BOSS factory scheduling system. A study comparing MICRO-BOSS against a macro-opportunistic scheduler suggests that the additional flexibility of the micro-opportunistic approach to scheduling generally yields important reductions in both tardiness and inventory.

Current research efforts include:

- Adaptation of MICRO-BOSS to deal with sequence-dependent setups
- Development of micro-opportunistic reactive scheduling techniques that will enable the system to patch the schedule in the presence of contingencies such as machine breakdowns, raw materials arriving late, job cancelations, etc.

APPENDIX: PROBLEM SETS

Problem Sets			
Number of Bottlenecks	Avg. Due Date	Due Date Range	Problem Set
1	loose	wide	1
1	loose	narrow	2
1	tight	wide	3
1	tight	narrow	4
2	loose	wide	5
2	loose	narrow	6
2	tight	wide	7
2	tight	narrow	8

REFERENCES

1. J. Adams, E. Balas, and D. Zawack. "The Shifting Bottleneck Procedure for Job Shop Scheduling". *Management Science 34*, 3 (1988), 391-401.

2. A.K. Mackworth and E.C. Freuder. "The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems". *Artificial Intelligence 25*, 1 (1985), 65-74.

3. Peng Si Ow and Stephen F. Smith. "Viewing Scheduling as an Opportunistic Problem-Solving Process". *Annals of Operations Research 12* (1988), 85-108.

4. P.S. Ow, S.F. Smith, and A. Thiriez. Reactive Plan Revision. Proceedings of the Seventh National Conference on Artificial Intelligence, 1988, pp. 77-82.

5. Norman Sadeh. *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling*. Ph.D. Th., School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, March 1991.

6. P. Serafini, W. Ukovich, H. Kirchner, F. Giardina, and F. Tiozzo. Job-shop scheduling: a case study. In *Operations Research Models in FMS*, Springer, Vienna, 1988.

Decision-Theoretic Control of EUVE Telescope Scheduling*

Othar Hansson and Andrew Mayer

Heuristicrats Research Inc.
1678 Shattuck Avenue, Suite 310
Berkeley, CA 94709-1631

Computer Science Division
University of California
Berkeley, CA 94720

Abstract

This paper describes DTS, a *decision-theoretic scheduler* designed to employ state-of-the-art probabilistic inference technology to speed the search for efficient solutions to constraint-satisfaction problems. Our approach involves assessing the performance of heuristic control strategies that are normally hard-coded into scheduling systems, and using probabilistic inference to aggregate this information in light of features of a given problem.

BPS, the Bayesian Problem-Solver [8], introduced a similar approach to solving single-agent and adversarial graph search problems, yielding orders-of-magnitude improvement over traditional techniques. Initial efforts suggest that similar improvements will be realizable when applied to typical constraint-satisfaction scheduling problems.

1 DTS Problem Domain and Representation

The Decision-Theoretic Scheduler, DTS, is designed for over-subscribed project scheduling problems. Although our work has not focused on problem representation, the probabilistic techniques used in DTS suggest the possibility of representing stochastic domains, in which, for example, task durations are variable and possibly inter-correlated. Primarily, work on DTS has focussed on search control, particularly through the combination of heuristic evaluation functions. As discussed below, this makes DTS a promising approach for new domains in which sophisticated domain-specific heuristic functions have not been developed. Finally, the utility-theoretic basis of DTS' optimization criteria makes it an attractive approach for problems in which complex tradeoffs must be made among competing tasks and expensive real-world and computational resources.

*This research was supported by the National Aeronautics and Space Administration under contract NAS2-13340.

The current DTS effort is specifically targetted toward experiment scheduling on orbiting telescopes. The initial application domain is the Extreme Ultraviolet Explorer (EUVE) [6], which observes in the wavelength range of 70 to 760 angstroms. The EUVE is operated by the NASA Goddard Space Flight Center and the Center for EUV Astrophysics (CEA) at the University of California, Berkeley. Although the remainder of the paper is concerned with the methodology underlying the system, we describe the problem briefly here.

The tasks in the EUVE scheduling problem are astronomical observations. Although an initial EUVE total sky survey employs fairly short observations, the observations we are concerned with are fairly long. In practice, however, observations will be broken into approximately 30 minute chunks by a variety of unavoidable and largely unpredictable interruptions.

The resources in this scheduling problem are observational instruments. Although the EUVE has several on-board instruments, we concentrate on scheduling guest observations, which are restricted to the EUV spectrometer instrument. Thus, we may consider this a single-resource scheduling problem.

The constraints in the problem are determined by the positions of observational targets, the position of the Observer platform, and the position of obstacles such as planets, the sun and atmospheric anomalies. There are few explicit inter-task constraints, aside from those derived from the time required to retarget the instrument.

1.1 Problem Representation

We phrase these scheduling problems in the language of constraint-satisfaction. Formally, a constraint-satisfaction processing (CSP) problem consists of a set of variables together with a set of constraints on the legal values of those variables. The CSP problem is

solved when the variables have been instantiated to a set of values that violate none of the constraints. A wide variety of problems can be phrased as CSP problems, including scheduling, graph-coloring, interpretation of visual scenes, etc. (van Hentenryck [23] provides a survey).

A large class of scheduling problems can be represented as constraint-satisfaction problems, by representing attributes of tasks and resources as variables. Task attributes include the scheduled time for the task (start and end time) and its resource requirements. The primary attribute of resources is availability or accessibility. A schedule is constructed by assigning times and resources to tasks, while obeying the constraints of the problem.

Constraints capture logical requirements: a typical resource can be used by only one task at a time. Constraints also express problem requirements: task T_x requires N units of time, must be completed before task T_y, and must be completed before a specified date. Both van Hentenryck [23] and Zweben et al. [26] provide concise illustrative examples of scheduling problems represented as CSP problems. For compatibility and evaluation purposes, our problem representation is based on that of the SPIKE system [15].

1.2 Optimization Criteria

DTS uses multiattribute utility functions to represent user preferences for both solution quality and computational costs. Multiattribute utility theory (MAUT) is a formalized method for quantifying preference relationships among a set of uncertain outcomes. The next section will describe the use of utility functions in search control, but one simple distinction can be made between DTS and traditional scheduling systems at this point. Most scheduling systems have a single vehicle – the heuristic evaluation function – for representing both search control knowledge and user preferences. This overlap of search control and schedule evaluation makes the task of constructing good heuristic functions more difficult than it needs to be.

For example, the ISIS [7] and SPIKE systems [15] employ suitability functions which state the "preferences" of the user as a function of a task and a time assignment. Consider the optimal schedule for a set of tasks. For the SPIKE and ISIS systems, the suitability function which best satisfies the user's preferences is that which "encodes" this optimal schedule by peaked 0/1 values. Such a suitability function essentially encodes search control information. Although this is an extreme case, suitability functions and other heuristic evaluation functions must encode both search control and schedule evaluation information. DTS separates heuristic functions and schedule evaluation, yielding a mathematically principled search algorithm, while at the same time simplifying the knowledge-engineering task for the designer of a new scheduling system.

2 DTS System Overview

There are numerous inadequacies with existing CSP technologies. Existing CSP heuristics, like all heuristic evaluation functions, are imperfect, and exhibit highly domain-specific performance. Although they often provide useful search control advice, they introduce uncertainty into the search algorithms which rely on them. However, CSP problem-solving paradigms do not address this issue because they fail to provide uncertainty models for heuristic information. Consequently, current techniques are forced to pay a large and unnecessary computational price in cases where the heuristic function makes incorrect classifications. Furthermore, the algorithms are doomed to repeat these costly mistakes forever, as there is no learning mechanism designed to improve a CSP heuristic's performance over time.

Existing heuristic functions confuse many different kinds of information. Some heuristic functions estimate the quality of the completion of a partial schedule. Others estimate the difficulty of finding a feasible solution. This confusion results in inadequate guidance for human experts who are charged with developing good heuristic functions. The development of a good heuristic often amounts to little more than parameter-adjustment to improve performance.

The problem of developing heuristics is compounded by the lack of technological developments which allow evidence from multiple heuristic functions to be combined. For this reason, the selection of appropriate heuristics and problem-solving techniques for any given CSP domain remains a craft despite years of comparative study.

DTS, which is derived from previous work on BPS (the Bayesian Problem-Solver) [8], is designed to address these problems. One area of innovation is the *heuristic error model*: a probabilistic semantics for heuristic information, based on the concept of conditional probability in statistical decision-theory [11]. Heuristics are interpreted by correlating their estimates with the actual payoffs of problem-solving instances. When a problem is solved, the heuristic error model is updated, adapting it to the problem's specific characteristics. Multiple heuristics are combined by correlating payoffs with a *set* of heuristic estimates. This provides a sound method for combining multiple heuristics.

This section describes the methodology which forms the core of DTS. In addition to the traditional tools developed for scheduling systems, the DTS approach relies heavily on technologies such as multiattribute utility theory [16; 25], Bayesian probabilistic inference [1; 3; 22], information-value theory [14; 19] and Bayesian learning [4; 17].

2.1 Decisions in Scheduling

DTS employs decision-theoretic techniques to guide the search for feasible and efficient schedules. Decision theory and its central maximum expected utility principle describe methods for making decisions when the outcomes of those decisions are uncertain. A scheduling system is just such a decision-maker. The decisions to be made by a scheduling system include:

1. Which portion of the search tree should be explored next?
2. Should search continue, or should the current best solution be output to the user?
3. If an infeasible schedule must be repaired, which set of repairs is best?

When considered in isolation, these decisions seem very difficult. In fact, they are difficult to formulate and solve in a sound and efficient manner. Existing search algorithms make these decisions in an *ad hoc* manner. Our approach is to apply the standard principles of rational decision-making to these decisions.

The theory of expected utility [24] claims that *rational* decision-makers attach *utilities* to all possible outcomes, and when faced with a decision under uncertainty, select that outcome with maximum expected utility. Utility is the subjective assignment of value to potential outcomes, when the exact outcome is uncertain. An extension of utility theory, which describes the behavior of a decision-maker faced with multiple, and possibly conflicting objectives, is multiattribute utility theory (MAUT).

Under certain natural restrictions on the consistency of a sequence of decisions (i.e., the axioms of decision theory), the fundamental theorem of decision theory states that a consistent decision-maker acts as if he were following the dictates of the theory: i.e., a utility function may be constructed to model his preferences and the MEU principle used to reproduce his decisions. Many artificial intelligence researchers have recently turned to decision-theoretic principles in attempting to engineer sound but resource-conscious systems.

2.2 Heuristic Error Models

The fundamental problem with prior work in scheduling is that the semantics of heuristic functions are defined only in terms of performance: heuristics are "magic" parameters that determine the speed of search. Not surprisingly, an expert's effort in development of scheduling systems is often dominated by time spent handcrafting a high-performance heuristic through parameter adjustment. Because the semantics of heuristics are unclear, even the most sophisticated combination and learning mechanisms are limited in their effectiveness.

DTS takes the approach that there are crucial quantities relating to a state in a search tree, i.e., the attributes of the utility function, including the cost of the search performed, whether a solution was found and the attributes which describe the solution quality. If those attributes were known, decision-making would be trivial. In DTS, heuristic evaluation functions are treated as evidence relating to the value of one or more of the utility attributes. We refer to the set of attributes as the *outcome* of that search tree node.

It is apparent that different heuristics serve to measure different attributes of utility (search cost, solution quality, solution probability). For example, a CSP heuristic such as "Most Constraining Variable" is implicitly encoding information about ease of search: a variable which heavily constrains unassigned variables will produce a smaller search tree. This intuition about the heuristic is borne out empirically.

This association between raw heuristic values and utility attributes is referred to as a *heuristic error model*. Briefly, the heuristic error model provides a simple means of infusing domain specific information into the problem-solving process by associating immediately visible features of a state with a belief about the outcome of that state. "Features" of the state S_i are indicated by a heuristic function, $h(S_i)$, and the association with outcome attributes A_i is provided by the heuristic estimate $\Pr\{h(S_i)|A_i\}$.

Learning Heuristic Error Models Historically, nearly all heuristic search algorithms have used the face-value principle of heuristic interpretation, i.e., behaving as if these estimates were perfect. As a result, most existing heuristic search algorithms violate the basic axioms of consistency and rationality in decision-making.

In contrast, DTS will gather statistics to calibrate the heuristic error model over time, as problems are solved. When introducing the system in a new domain, a *prior* probability distribution will be fine-tuned based on "training exercises" with representative problems. This calibration process will improve DTS performance, tailoring it to the characteristics of real-world problems as they are encountered. When the heuristic function is imperfect, DTS will learn a mapping which "corrects" the heuristic to as great a degree as possible. Finally, the DTS learning capability will reduce the burden on human experts to produce highly complex heuristics. Their experience can be encoded as a default initial belief, or *prior probability*.

Combining Heuristics In our initial experiments in scheduling, a primary advantage of the heuristic error model has been the ability to combine multiple heuristics. Artificial intelligence techniques have never offered powerful methods for combining heuristics. A popular approach is to handcraft a composite heuristic

which is a linear combination of individual features.

By combining multiple heuristics, DTS isolates measurements of the difficulty and promise of completing potential assignments. Hence, DTS will make use of heuristics which previously have led to inconsistent performance: if there are any easily characterized contexts (in terms of other features) in which the heuristic performs well, DTS will recognize that fact. This context-dependency of heuristic functions has long been recognized in other search applications such as game-playing.

2.3 Use of Heuristic Error Models

The DTS architecture relies on the Bayesian network data structure [18], the primary artificial intelligence tool for representing and reasoning with probabilistic information. The Bayesian network is used to provide information for decisions such as the most promising region of the search tree to expand next, and the most promising schedule extension or modification to choose next. As described below, Bayesian networks can integrate a variety of information in the service of such decisions, including multiple heuristic evaluation functions and the search tree's topology.

The nodes of a Bayesian network are variables which represent the attributes of the domain. The arcs of the network connect dependent variables, representing relationships among domain attributes. Dependencies can be due to functional, causal or correlative relationships among variables. Dependencies between variables in the network are encoded in a modular fashion by specifying a conditional probability distribution for each network node conditioned on the values of its parents. Although most work has centered on the discrete variable case, Bayesian networks can also incorporate continuous variables.

The variables in the DTS Bayesian network are of two types. One type are variables which represent the multiattribute outcomes (e.g., schedule cost, search cost) of legal partial assignments in the CSP problem's state-space. The other type of variables represent the values of heuristic evaluation functions, the primitive feature recognizers of the domain (e.g., the number of remaining values, the degree of the constraint graph). The structure and semantics of the Bayesian network are described in detail in [12].

The dependency structure and parameters in a Bayesian network enable the efficient computation of the *joint probability* of any instantiation of variables. A fundamental theorem of probability theory indicates that from a joint probability distribution, and thus from a Bayesian network, any well-formed probabilistic question (i.e., conditional probability) can be answered, by the application of Bayes' rule and marginalization. Common queries are of the following form:

- The "next-best-test," or most crucial piece of evidence to gather. In search, this corresponds to the area of the search tree which is most crucial to explore next.
- The conditional probability of a variable instantiation, given the available evidence. In search, such probabilities can be used together with a utility function for maximum-expected-utility decision making, including the choice of task assignments, schedule modifications, etc.
- The most likely instantiation of all variables, given the available evidence. In search, this can be used as a simplified "situation assessment" of the state of the search.

2.4 Utility-Directed Selective Search

DTS employs advanced decision theory techniques to direct its search process. Decision theory, together with the probabilistic inference machinery described above, enables DTS to determine the best portion of the search tree to explore next. In addition to the obvious improvements in search efficiency, this facility improves the flexibility of DTS. By altering the utility function provided as input to the system, DTS may be tailored to trade off increased search time for increases in schedule quality, or to produce schedules with different desirable attributes. For reactive scheduling applications, alterations to the existing schedule can be given negative utility, in which case DTS will avoid them where possible.

The essence of decision-theoretic search control is the realization that there is quantifiable value in the acquisition of information. It should be clear that some pieces of information are more valuable than others. In addition, the acquisition of information has costs – in scheduling search, this cost is increased computation time. If these computations squander time and other resources, the solution may be found too late to be of use. If these computations are neglected, a poor solution may be found. However, if these computations are chosen wisely, the system will provide high quality solutions despite limited computational resources. Decision theory has spawned a subfield known as *information value theory* which deals with the issue of *deciding*, at a metalevel, what information to acquire in order to make better decisions at the base level.

Decision-theoretic search control thus involves the isolation of decisions that are made in the course of search, and applying the techniques of decision theory to make *rational* decisions at these choice points. The decisions made in heuristic search include choices among possible search tree expansions and possible heuristic evaluations. DTS applies information value theory by using the maximum expected utility criterion to control its information-gathering search.

Such decisions form the basis of a selective search algorithm that explores the search tree in a nonuniform manner so as to find a high quality solution in as little

time as possible. In simple terms, rather than being a "depth-first" or "breadth-first" search, DTS exhibits a "highest-utility-first" search behavior, gathering information most relevant to the decisions that must be made.

An additional benefit of the decision-theoretic search control is that search control and heuristic information are represented in a declarative manner. As different search spaces (e.g., partial schedules, complete but infeasible schedules, etc.) correspond to different decision problems, DTS can be applied to any search space. In the prototype system, the techniques have been applied to search through the space of valid partial schedules.

2.5 DTS Version 1.0

The DTS prototype employed a simplified decision-theoretic control mechanism which was adapted to a conventional backtracking search algorithm: this allowed for controlled experiments on DTS vs. traditional algorithms.

The only search control decisions made in traditional backtracking systems are the selections of which subtrees of the search graph to explore next. Once a subtree is selected (by selecting the next variable or value), it is explored exhaustively unless a solution is found. Such an ordering problem can be viewed as a decision-tree. Figure 1 depicts the choice of ordering two subtrees A and B. A simple theorem [12] shows that the system's expected utility (search time to first solution) is maximized if variables (or values) are ordered by the quantity $P(v)/C(v)$, where $P(v)$ indicates probability of finding a solution in the subtree, and $C(v)$ indicates the cost of searching the subtree (whether or not a solution is found). $P(v)$ and $C(v)$ are attributes of the payoff mentioned above. The experiments described in the next section confirm that once $P(v)$ and $C(v)$ are learned, this rule outperforms traditional backtracking search algorithms which interpret heuristic estimates at face value. This result indicates that decision-theoretic search-control improves overall system performance. A similar analysis can also be performed for iterative improvement [12].

We note here that while heuristics are usually very good at rank-ordering nodes based on either $P(v)$ or $C(v)$ individually, the rank-ordering for the combination is typically incorrect. DTS' heuristic error model corrects for this.

For clarity, we summarize the prototype implementation here. The prototype performs a backtracking search, using the standard optimizations of forward-checking and dynamic search rearrangement. The search is ordered by the expected utility selection criteria ($P(v)/C(v)$) discussed above. The estimates of $P(v)$ and $C(v)$ are derived from the heuristic error model, using traditional CSP heuristics. The heuristic error model is updated during and between trials

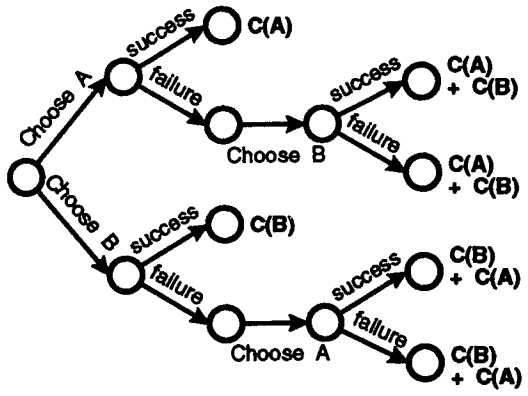


Figure 1: Decision Tree for Value-Ordering Problem (Values A and B)

using a bucketed histogram, and interpreted by Laplacian estimation.

2.6 Performance

Space limits us to a discussion of only three aspects of the DTS prototype's performance characteristics: combination of heuristics, learning heuristic error models, and generalizing learned information.

Combining Heuristics The primary strength of the DTS prototype is the method for combining information from separate heuristic evaluation functions to improve constraint-satisfaction search control. Experiments with the prototype on the Eight Queens and Bridge-Construction Scheduling [23] problems confirm that the combination of heuristic functions provides more information than any of the heuristics taken individually. This translates into significant reductions in overall search time.

Traditionally, CSP algorithms make use of a variable ordering heuristic and a value ordering heuristic. Figure 2 shows the performance of a standard CSP algorithm using all possible pairings (A1, A2, B1, B2) of two well-known variable ordering heuristics (Most Constraining Variable (A), Minimum Domain Variable (B)) and two well-known value ordering heuristics (Least Constraining Value (1), Dechter's Value Heuristic (2)[2]). Also shown is the DTS prototype (DTS-Joint), which dominated the competition by using all four heuristics in combination. The horizontal axis plots the number of problem instances solved and the vertical axis plots the running average of search time over the entire experiment. The plot, but not the average, begins with the tenth problem instance.

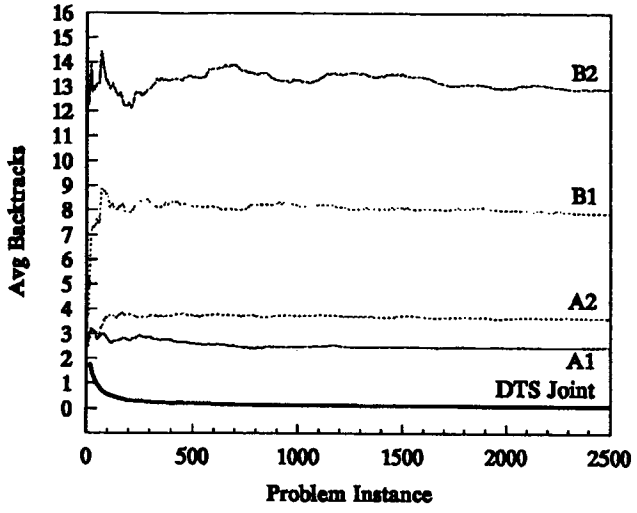


Figure 2: Eight Queens: Combining Heuristics vs. Heuristics in Isolation

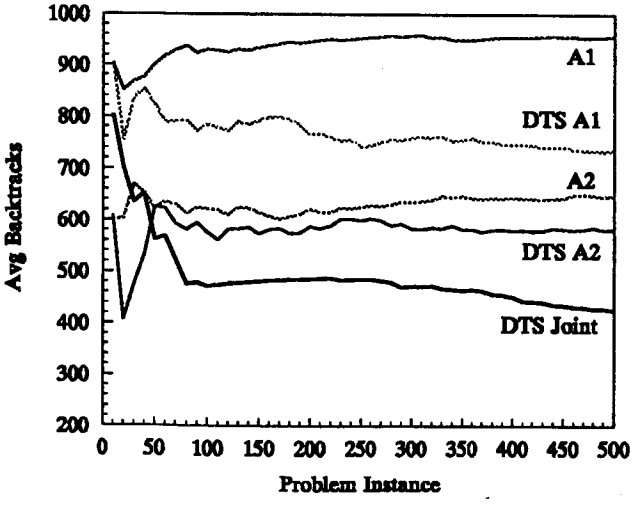


Figure 3: Bridge-Construction Scheduling: Combining Heuristics vs. Heuristics in Isolation

Figure 3 shows a corresponding graph for the Bridge-Construction Scheduling problem. The variable ordering heuristic used was Minimum Domain Variable and the value ordering heuristics were Least Constraining Value (curve A1) and ASAP, "as soon as possible" (curve A2). Also shown are the corresponding individual DTS performance curves (DTS A1, DTS A2) as well as the combined heuristic performance curve (DTS-Joint).

To summarize both graphs, the improvement is seen to be nearly 50% on average for Bridge Construction Scheduling, and over 95% for the Eight-Queens problem. Note that the sharp downward slope of the DTS-Joint *running average* in Figure 3 demonstrates the performance improvement accrued by learning, unattainable using traditional techniques.

Learning Heuristic Error Models Figure 4 displays an example heuristic error model learned over the course of 2500 Eight-Queens problem instances (for the Minimum Domain heuristic). The horizontal axis plots the heuristic function estimate and the vertical axis plots the preference for that estimate. In DTS, preference is based upon the expected utility associated with a heuristic estimate (dashed line). In traditional algorithms, the heuristic is assumed to rank-order alternatives perfectly, and therefore, preference is a monotonic function of the heuristic estimate.

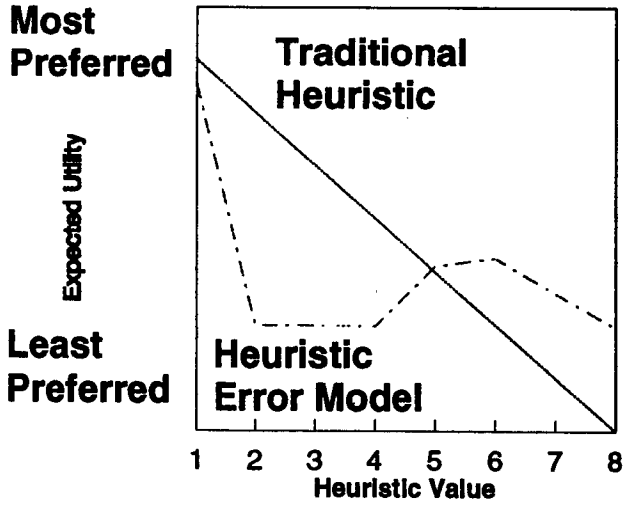


Figure 4: Sample Heuristic Error Model

The discrepancy between the heuristic estimates and the actual utilities explains the poor performance of traditional approaches, which assume perfect heuristic estimates. Further, it explains why DTS outperforms these techniques, as it does not make this assumption, and instead learns to correct for the discrepancy.

Bootstrapping Learned Information An additional benefit of the heuristic error model is the ability to generalize learned data across domains. For example, Figure 5 depicts the performance of DTS on the Thirty-two-Queens problem with 1) no prior heuristic error model, and 2) a heuristic error model generalized (or "bootstrapped") from the 2500 Eight-Queens examples solved in Figure 2. Generalizing data from the simpler domain has reduced search complexity. This is particularly important as the time required to calibrate heuristic error models increases with problem complexity.

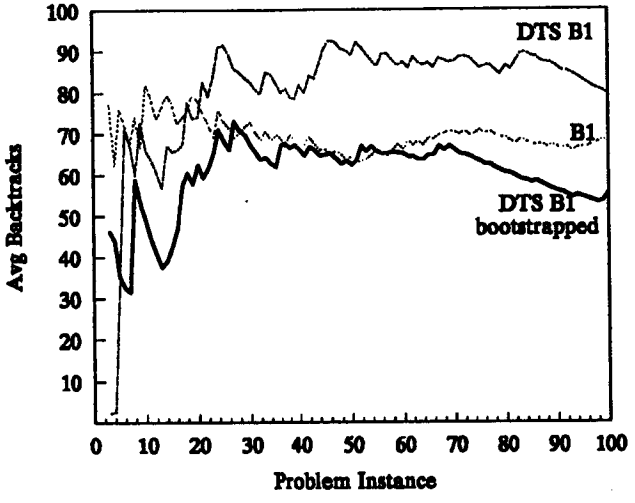


Figure 5: Generalizing Data to Larger Domains

3 Related Work

The DTS system is based on the authors' previous work on the Bayesian Problem-Solver (BPS) system. BPS has been applied to classic AI problem-solving [8], game-playing [10] and planning [9] domains. Similar decision-theoretic approaches are being considered for applications to other complex multiattribute optimization problems.

This work is most closely related, in assumptions and techniques, to the recent work in applying decision theory to problems such as medical diagnosis [13] and image interpretation [5]. Others have applied decision theory to heuristic search applications. These researchers have typically limited themselves to grafting decision-theoretic principles onto existing algorithms[20; 17]. Like the decision-theoretic backtracking discussed above, this makes for an interesting starting point, and there are many more interesting possibilities to explore in the future.

Given its probabilistic basis, this work might be assumed to be related to systems designed for stochastic scheduling problems. Unfortunately, we have not had an opportunity to consider stochastic problems as of yet, although we anticipate that the probabilistic representation and inference mechanisms in DTS will ease the transition to stochastic problems.

4 Conclusions

The use of Bayesian probability theory in DTS underscores that scheduling involves decision-making under uncertainty, and illustrates how imperfect information can be modeled and exploited. The use of multiattribute utility theory in DTS underscores that scheduling involves complex tradeoffs among user preferences. By addressing these issues, DTS has demonstrated promising performance in preliminary empirical testing.

References

[1] R. T. Cox. Probability, Frequency and Reasonable Expectation. *American Journal of Physics*, vol. 14, 1946.

[2] R. Dechter and J. Pearl. Network-Based Heuristics for Constraint-Satisfaction Problems. In *Search in Artificial Intelligence*, L. Kanal and V. Kumar, eds., Springer-Verlag, New York, 1988.

[3] B. de Finetti. *Theory of Probability.* John Wiley, New York, 1974.

[4] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, New York, 1973.

[5] G. J. Ettinger and T. S. Levitt. An Intelligent Tactical Target Screener. In *Proceedings of the 1989 DARPA Image Understanding Workshop.* Morgan Kaufmann, San Mateo, CA, 1989.

[6] The EUVE Guest Observer Center. *EUVE Guest Observer Program Handbook.* Appendix G of NASA NRA 92-OSSA-5. Center for EUV Astrophysics, Berkeley, January 1992.

[7] M. S. Fox. *Constraint Directed Search: A Case Study in Job-Shop Scheduling.* Pitman, London, 1987.

[8] O. Hansson and A. Mayer. Heuristic Search as Evidential Reasoning. In *Proceedings of the the Fifth Workshop on Uncertainty in Artificial Intelligence*, Windsor, Ontario, August 1989.

[9] O. Hansson, A. Mayer, and S. J. Russell. Decision-Theoretic Planning in BPS. In *Proceedings of the AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*, Stanford, March 1990.

[10] O. Hansson and A. Mayer. A New and Improved Product Rule. In *Proceedings of the the Eighth International Congress of Cybernetics & Systems*, New York, June 1990.

[11] O. Hansson and A. Mayer. Probabilistic Heuristic Estimates. *Annals of Mathematics and Artificial Intelligence*, 2:209–220, 1990.

[12] O. Hansson and A. Mayer. Decision-Theoretic Control of Artificial Intelligence Scheduling Systems. HRI Technical Report No. 90-1/06.04/5810, September 1991.

[13] D. E. Heckerman. *Probabilistic Similarity Networks.* MIT Press, Cambridge, 1991.

[14] R. A. Howard. Information Value Theory. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SSC-2, 1965.

[15] M. D. Johnston. Knowledge-Based Telescope Scheduling. in *Knowledge-Based Systems in Astronomy*, A. Heck and F. Murtagh, eds., Springer-Verlag, New York, 1989.

[16] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley, New York, 1976.

[17] K.-F. Lee and S. Mahajan. A Pattern Classification Approach to Evaluation Function Learning. *Artificial Intelligence*, vol. 36, 1988.

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo, CA, 1988.

[19] H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory.* Harvard University, 1961.

[20] S. J. Russell and E. Wefald. Principles of Metareasoning. In *Proceedings of the 1st Conference on Principles of Knowledge Representation and Reasoning.* Toronto, 1989.

[21] N. Sadeh. Lookahead Techniques for Activity-Based Job-Shop Scheduling. Technical Report TR CMU-RI-TR-89-2, CMU, the Robotics Institute, 1989.

[22] L. J. Savage. *The Foundations of Statistics.* Dover, New York, 1972.

[23] P. van Hentenryck. *Constraint-Satisfaction in Logic Programming.* MIT Press, Cambridge, MA, 1989.

[24] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[25] D. von Winterfeldt and W. Edwards. *Decision Analysis and Behavioral Research.* Cambridge University Press, 1986.

[26] M. Zweben, M. Deale and R. Gargan. Anytime Rescheduling. in *Proceedings of the DARPA Planning Workshop*, Morgan Kaufmann, San Mateo, CA, 1990.

Massively Parallel Support for a Case-based Planning System

Brian P. Kettler James A. Hendler William A. Andersen

Department of Computer Science
University of Maryland
College Park, MD. 20742

Abstract

Case-based planning (CBP), a kind of case-based reasoning, is a technique in which previously generated plans (cases) are stored in memory and can be reused to solve similar planning problems in the future. CBP can save considerable time over generative planning, in which a new plan is produced from scratch. CBP thus offers a potential (heuristic) mechanism for handling intractable problems. One drawback of CBP systems has been the need for a highly structured memory to reduce retrieval times. This approach requires significant domain engineering and complex memory indexing schemes to make these planners efficient.

In contrast, our CBP system, CaPER, uses a massively parallel frame-based AI language (PARKA) and can do extremely fast retrieval of complex cases from a large, unindexed memory. The ability to do fast, frequent retrievals has many advantages: indexing is unnecessary; very large casebases can be used; memory can be probed in numerous alternate ways; and queries can be made at several levels, allowing more specific retrieval of stored plans that better fit the target problem with less adaptation. In this paper we describe CaPER's case retrieval techniques and some experimental results showing its good performance, even on large casebases.

1. INTRODUCTION

A case-based planning system solves planning problems by making use of stored plans that were used to solve analogous problems. Case-based planning (CBP) is a type of case-based reasoning (CBR), which involves the use of stored experiences ("cases"). There is strong evidence that people frequently employ this kind of analogical reasoning (e.g., (Vosniadou and Ortony 1989; Gentner 1989; Ross 1989)).

CBP systems consist of a planner and a casebase holding cases, which typically include a description of a planning problem (an initial state and goals to be achieved) and the plan(s) used to solve it (a sequence of primitive, executable actions). When a CBP system solves a new planning problem, the new plan is added to its casebase for potential re-use in the future. Thus the system learns from experience. Since feedback from the new plan's execution is also typically stored as part of the new case, the system can avoid any repeating failures it encountered.

CBP planners can be distinguished from generative planners, such as NONLIN (Tate 1976), which generate a plan from scratch by searching a space of partial plans. These systems expand a partial plan by adding actions to it and then checking the plan for helpful and harmful interactions between actions in the expanded plan — a costly process. CBP systems, in contrast, do not begin from scratch and attempt to find fully-instantiated plans with any harmful interactions already removed.

Most CBP systems use a version of the following process when given a new planning problem to solve: (1) retrieve case(s) from memory that are analogous to the current ("target") problem, (2) select one or more of these candidate cases that are most similar to the target problem, (3) adapt the selected cases (plans) to a new plan for the target problem, (4) get feedback on the new plan from its execution, (5) diagnose and repair any faults in the plan found during its execution, (6) store the repaired plan and the failure/repair information in the casebase.

The CaPER system (Cased-based Planning, Explanation, and Repair) is a case-based planner that is being developed to take advantage of the efficiencies of plan re-use while addressing some of the problems and limitations of case-based planners that use serial retrieval procedures on an indexed memory. To provide fast retrieval of cases (and other kinds of episodic knowledge) from a large, unindexed memory, CaPER makes use of the massive parallelism of the Connection Machine (CM) (Hillis 1985). The ability to quickly access memory permits frequent memory accesses. The implications of being able to go to memory often are great, and CaPER is designed to take advantage of this wherever possible.

This paper describes some of CaPER's components and the implications of fast, frequent memory access on their design. Section 1.1 describes some of the problems with the approach to case retrieval taken by most "traditional" CBP systems. Section 2 describes CaPER's memory and the PARKA system used to implement it. Section 3 describes CaPER's plan retriever. Section 4 will describe some promising empirical results of the

retriever for problems in the simplified automobile assembly domain being used to test the CaPER prototype. Section 5 briefly describes the CaPER components yet to be implemented and other future work. Section 6 describes some other systems that make use of massive parallelism for memory retrieval and compares those approaches with CaPER's.

1.1 Case Retrieval in "Traditional" Case-based Planning Systems

Most case-based systems use serial procedures to retrieve a set of cases from memory that are analogous to the target problem — e.g. the CHEF system (Hammond 1990a, 1990b). Features of the target problem are used as a probe to match against stored cases. Features are objects, object attributes, and relations between objects in the description of the target problem's initial situation and goals.

In order to provide more efficient case retrieval, the technique of indexing is used to restrict the kinds of features that may be used as part of a memory probe. It would be prohibitively expensive to sequentially compare a probe against each case in an entire casebase of realistic size (i.e., hundreds or thousands of cases). Indexing makes this approach more tractable by restricting the search for old cases to only parts of a casebase. When indexing is used, a stored plan can only be retrieved through a subset of its features or some abstraction of these features. For example, CHEF stores a plan for cooking beef and broccoli stir-fry under the indices "beef" and "broccoli" and abstractions "meat" and "vegetable". The abstractions used need to be limited to a few for efficiency's sake.

Choosing an indexing scheme that can provide efficient retrieval of relevant cases from memory is a difficult task. Indexing schemes are often domain-specific and task-specific and thus limit the general-purpose utility of the memory. As an alternative to the explicit indexing of cases, a domain theory might be used to limit the search of case memory. The use of a highly indexed memory or a domain theory limits the flexibility of memory retrieval.

Using a highly indexed memory or a strong domain theory sharply constrains case retrieval. A case can only be retrieved through its indices. Cases that share unindexed features with the target problem will not be retrieved. For example, one might have limited time to devote to cooking and thus desire a cooking plan for cooking any dish that takes less than 15 minutes to prepare. CHEF could not cope with such a query since it does not index plans by the time they take. It therefore would have to resort to checking potentially all cases in its casebase. One could add an index for time to each case in CHEF's casebase. But to do this for many features would lead to a proliferation of indices which could defeat the purpose of indexing, namely efficiency. As another example, one might be doing the cooking in a friend's kitchen that lacks a wok. Thus the query might be to find a cooking plan that does not require a wok. It is unlikely that cases would be indexed under each item that they do not use.

Indexing schemes typically impose an organization on memory. For example, a discrimination network is often used to organize cases in the casebase. Cases with similar indices are stored in the same subtree of a discrimination tree. A new case's indices typically must be computed when the case is stored in the casebase since a case needs to be placed under the right node of the discrimination network. Thus the designer of the indexing scheme needs to anticipate all the ways a case might usefully be retrieved in the future.

The computation of indexes can be expensive. One has to determine the indices of a new case at case storage time and the indices of the target problem that comprise the memory probe at case retrieval time. Finally, the psychological plausibility of indexing is questionable given that human memory access is parallel and associative with many remindings being generated: e.g., (Waltz 1989; Thagard and Holyoak 1989; Gentner 1989).

1.2 Overview of CaPER

CaPER uses a planning cycle similar to that of most CBP systems: (1) the Plan Retriever component retrieves, via the Episode Retriever component, plans from memory used to solve analogous planning problems; (2) the Plan Adapter/Modifier adapts these plans to produce a new plan for the target problem; (3) the Plan Tester presents the new plan to a user who supplies feedback from its execution; (4) the Plan Failure Diagnoser attempts to find the cause of any faults found in the plan and calls the Plan Adapter/Modifier to fix the plan and the Plan Tester to test the repaired plan; (5) the Episode Storer stores a successful plan is stored (along with execution feedback) as a new case in the casebase.

The Memory, Plan Retriever, and Episode Retriever components have been implemented. The Plan Adapter and Episode Storer components are currently under development. The remaining components are in the design phase.

2. CaPER's MEMORY

CaPER's memory includes both episodic and conceptual knowledge. Episodic knowledge describes particular, dated experiences that the system has had (or has been told of). Every episode is associated with a particular date, time, place, and other aspects of the situation in which it occurred. CaPER can apply planning episodes (cases), failure episodes, and repair episodes from particular situations to analogous situations it encounters in the future. A case is a kind of episode that describes a planning experience. This description includes the planning problem (initial situation and target goals), the plan(s) generated, the plan(s) actually instantiated during plan execution, and feedback on the success of the executed plan(s).

A plan that has been instantiated in a particular situation is termed an "e-plan" in CaPER and is also an episode. Plans consists of a hierarchy of component plans ("subplans"), each solving a goal/subgoal. For example, plan Build-Car-1 (used in Case-Build-Car-1) consists of subplans Build-Body-1, Install-Engine-1, Install-Sunroof-1, etc. These subplans can have component plans of their own. At the bottom of a plan hierarchy are primitive actions. These actions that have been instantiated as part of an e-plan are episodes themselves and are termed "e-actions". For example, e-action P-Install-Sunroof-1 is a primitive action of subplan Install-Sunroof-1 (and its parent plan Build-Car-1). A top-level plan, a plan that is not a component of another plan, is termed a "root" plan (e.g., Build-Car-1).

In addition to episodes, CaPER's memory includes conceptual knowledge: representations of concepts. Concepts are organized in generalization taxonomies, part/whole taxonomies, and in other relationships. Concepts include general concepts (e.g., taxonomies of physical things and abstract things), planning concepts (e.g., taxonomies of action conditions and effects), and domain concepts (e.g., a taxonomy relating car types and car parts and a taxonomy of types of car assembly actions).

Individuals are instances of concepts that participate in situations (real or hypothetical) and episodes. For example, car-1 is an instance a class whose members are denoted by the concept Car. Car-1 is a particular car with particular properties that here was the product of a planning episode (Case-Build-Car-1), the result of executing some plan (e-plan Build-Car-1). (Note that a numerical suffix is used to distinguish names of episodes and individuals from names of concepts).

CaPER's memory holds episodic and conceptual knowledge in a semantic network. Each node corresponds to an episode, concept, or individual. Nodes are linked by arcs that represent is-a, part/whole, and other kinds of relations. A piece of CaPER's memory is shown in figure 1. This includes a case (Case-Build-Car-1) of planning to build a car with a sedan frame, 4 cylinder engine, hard top, A/C, and sunroof. A plan (e-plan Build-Car-1) was used to build a car of this type in the Detroit factory, on May 6, etc. With inheritance mechanisms, plans can inherit goals and other properties from their parent plan or case. For example, plan Build-Car-1 (and all of its subplans) inherits the context feature "place Detroit-Factory" from its parent case, Case-Build-Car-1.

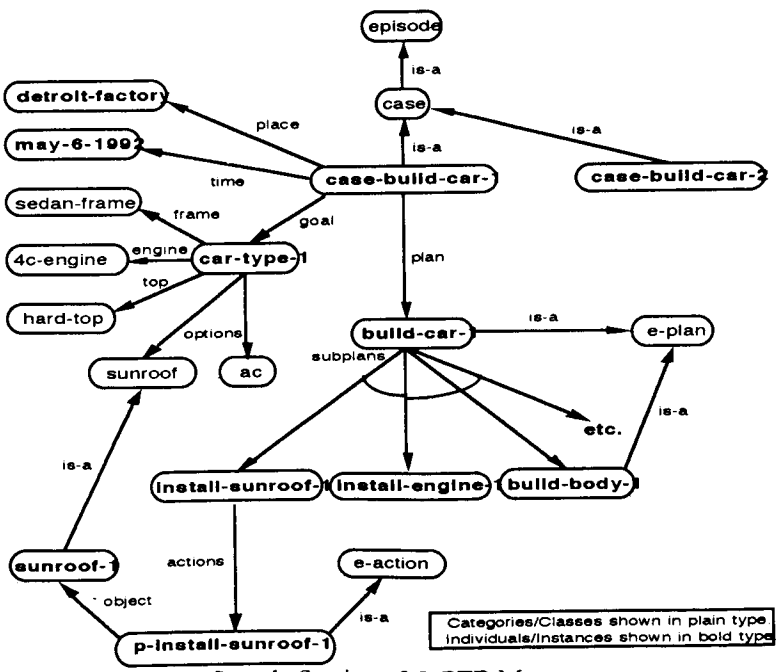


Figure 1. Sample Section of CaPER Memory

2.1 PARKA

CaPER's case memory is built on top of the PARKA knowledge representation system (Spector et al. 1990; Evett et al. 1992). PARKA is a frame-based system implemented on the CM-2 SIMD parallel supercomputer. It provides a rich representation language, consisting of concept descriptions and relations on those descriptions. In addition to representing concepts as a collection of relations with multiple inheritance, PARKA has mechanisms for easily describing set-theoretic and partonomic relations among concepts. PARKA is optimized for the performance of fast "recognition" queries of the form "all X's with features Y, Z, W..." in time proportional to the depth of the knowledge base (KB) — specifically O(D+m), where D is the depth of the KB and m is the number of conjuncts in the query. Typical queries of this type are answered in a fraction of a second (order 10 msec.), even for KBs in the tens of thousands of frames. Since the retrieval times are sensitive only to the depth of the KB, the retrieval algorithms are largely insensitive to KB size (Evett et al. 1992).

Concepts and individuals in CaPER's memory are represented using PARKA category frames and individual frames, respectively (see (Spector et al. 1992)). To extend the use of PARKA for work with CaPER, a "structure matcher" has been implemented, based on the conjunctive retrieval algorithm. This facility allows cases to be retrieved on arbitrary features, not restricted to any set of predetermined indexes. Retrieval probes are specified as a constraint graph which is matched against the entire KB simultaneously. A retrieval probe consists of a graph (V,R), where V is a set of variables, quantified over concepts in the KB, and R is a set of relations which must hold simultaneously between elements of V. The algorithm returns all such satisfying assignments. Preliminary experiments with the initial implementation of the structure matcher have demonstrated retrieval times under a second for a complex probe against a case base with hundreds of cases. Future implementations are expected to reduce retrieval times to under 100 msec. for comparable probes.

3. THE CaPER PLAN RETRIEVER

The design of the Plan Retriever exploits CaPER's ability to do fast, and hence frequent, memory accesses. Here the ramifications of this ability are twofold. First, CaPER can retrieve cases from memory that share any features in common with target problem. Second, CaPER can efficiently do multiple plan retrievals to get different plans (or pieces of plans), each of which can be used to solve a different part of the target problem.

With indexed memories, a case can only be retrieved through its indices, which are typically fixed at case storage time. Since it uses parallel procedures for retrieving episodes, CaPER does not need to use indexing, which serial retrieval procedures require for efficiency. CaPER's memory is therefore unindexed, and episodes can be retrieved on potentially any combination of their features. In CaPER, episodes are interconnected semantic networks of their constituent concepts. Thus it possible to access an episode through any of these constituent concepts. Furthermore, it is possible to issue highly specific queries of CaPER's memory since any feature can be a part of the memory probe.

The use of highly specific queries is facilitated by the ability to do many such queries cheaply. When looking for an episode analogous to a target, CaPER can issue highly specific queries and, if they fail, more general ones. If a more specific query is successful, fewer episodes will typically be retrieved, and there will be less work (in serial) to chose the best from among them (i.e., the ones most similar to the target). A more general query will typically result in more candidate episodes being retrieved and more work being required to evaluate them.

Another ramification of doing frequent memory accesses is being able to retrieve multiple different plans from memory, each of which solves a part of the target problem. CBP systems that use serial procedures typically go to memory infrequently because of the high cost of doing so. Thus they retrieve only one old case/plan (or in some systems, a few old cases) to adapt to solve all of the goals of the target problem. In contrast, CaPER can quickly retrieve several plans (or subplans) to achieve different goals of the target problem and merge them with the result being a better-fitting, composite plan that solves all (or most) of the target goals.

Massive parallelism also makes PARKA's retrieval algorithms run in time virtually independent of the size of the memory (see section 2.1). This means that the techniques used by CaPER/PARKA can scale up to very large memories with thousands of cases or more. In contrast, CBP systems that use serial procedures can typically provide reasonable performance on a casebase with at most tens of cases, unless highly restrictive indexing schemes (or domain theories) are used.

3.1 The Plan Retrieval Process

The CaPER Plan Retriever attempts to retrieve one or more e-plans from memory that solve one or more of the target goals (or goals similar to the target goals). The Plan Retriever also looks for old plans that had initial situations similar to the initial situation of the target problem. The old plans retrieved from memory may be whole plans or plans that are components of other plans. The hierarchical organization of plans in CaPER's memory enables component plans to be retrieved out of larger plans.

The Plan Retrieval Process is controlled by the Plan Retriever and contains these (high-level) steps:

1. Attempt to retrieve an old ("source") plan belonging to a case in memory that is most similar to the entire target problem (the target goals plus the initial situation). This plan is termed the "root source plan". If this step fails, CaPER cannot proceed since it must find at least one old plan to be the basis for the new plan. A generative planner could then be invoked in this event.
2. For each target goal/subgoal that the root source plan does *not* achieve, attempt to retrieve a source plan/subplan that is most similar to the target subproblem (the target goal plus the initial situation). If this step fails for a target goal, CaPER could use a generative planner to create a plan to achieve the goal.
3. For the root and other source plans retrieved, invoke the Plan Adapter to adapt each plan to the target problem and merge the adapted plans.

An example of this process in operation is given in Section 3.1.1.

In Steps 1 and 2, an attempt is made to find plans to build a similar car in a similar situation. Plans are retrieved by probing memory via the Episode Retriever which in turns calls PARKA's Structure Retriever. Memory is probed using the most important ("key") features of the target problem with respect to the domain (since memory is unindexed, any feature can potentially be a key feature). Key features typically include the target goals and relevant features from the initial situation of the target problem. For the simplified automotive assembly domain, the types of key features in order of importance are: type of frame (e.g., "sedan frame"), type of top (e.g., "hard top"), type of engine (e.g., "8 cylinder turbo engine"), type of options (e.g., "sunroof"), and certain features of the initial situation (e.g., "Detroit factory"). A feature is considered more important if it has a greater impact in generating a plan.

In the event that no plans are retrieved with the probe, the features it contains are generalized in order of increasing importance using taxonomic information from memory, and memory is probed again. If a probe succeeds, the cases (or parts of cases) that were retrieved with it are ranked using a similarity metric. A similarity metric is a measure of the goodness of the analogy between an old case and the target problem/case.

CaPER's similarity metric is based on how many key features the cases share (at some level of generality). For each feature of a retrieved case that matches, at some level of generality, a key feature of the target problem, the old case receives 1 point × 1/match-generality. An exact match of features (e.g., "8 cyl. turbo engine" with "8 cyl. turbo engine") is a match generality of 1. For each level difference between the features in the generalization hierarchy, the match generality increases by 1. For example, "8 cyl. turbo engine" with "8 cyl. engine" is a match generality of 2 (since the latter is a parent — one level above — the former) and a score of $1 \times 1/2 = 0.5$.

The effect of using match generality is to give more weight to more specific matches. The advantage of using a simple similarity metric, which CaPER currently computes serially for each case returned, is its ease of computation. Whether this simple similarity metric results in the best ranking of cases is still an open question to be resolved by experimentation.

3.1.1 An Example of the Plan Retrieval Process

To test the CaPER prototype, we have developed a simplified automobile assembly domain. CaPER's initial casebase consists of plans for building cars that were generated using our implementation of Tate's NONLIN planner.

In this example, the target problem has a goal to build a car with a sedan frame, hard top, 8 cylinder turbo engine, A/C, and a sunroof. (It is assumed that the car is being built by hand/robot rather than using an assembly line so that the order of steps may vary from car to car). The target problem also has an initial situation: the car is to be built at the Detroit factory, on a particular date, etc.

In Step 1, memory is first queried using the probe: <sedan frame, hard top, 8 cyl. turbo engine>. This probe fails, and the feature "8 cyl. turbo engine" is be generalized to "8 cyl. engine" (using the knowledge from the concept taxonomy that an 8 cyl. turbo engine "is-a" 8 cyl. engine). The probe <sedan frame, hard top, 8 cyl.

engine> fails, and "8 cyl. engine" is generalized (maximally) to "engine". If the probe <sedan frame, hard top, engine> fails then the next least important key feature, "hard top", would be generalized (to "top"). Fortunately, this probe does not fail and the following old cases are retrieved:

1. **Case: Case-Build-Car-1; E-Plan: Build-Car-1;**
   Car: Car-1 (sedan frame, hard top, 4 cyl. engine, A/C, sunroof, luggage rack); Location: Detroit-factory; Plan Status: Successfully executed; Similarity Score: 5 1/3.
2. **Case: Case-Build-Car-4; E-Plan: Build-Car-4;**
   Car: Car-1 (sedan frame, hard top, 4 cyl. engine, A/C, sunroof, luggage rack); Location: Seattle-factory; Plan Status: Successfully executed; Similarity Score: 4 1/3.
3. **Case: Case-Build-Car-8; E-Plan: Build-Car-8;**
   Car: Car-1 (sedan frame, hard top, 4 cyl. engine, A/C, luggage rack); Location: Seattle-factory; Plan Status: Successfully executed; Similarity Score: 3 1/3.

The similarity score for each case/plan retrieved is computed. While plans Build-Car-1 and Build-Car-4 both achieve the same goals, they differ as to location where the car was built. Since the target car is to be built in Detroit, plan Build-Car-1 will be preferred. The location might be important in choosing between the plans since the available machines, tools, and expertise differ between locations. Plan Build-Car-1 is selected as the root source plan. It contains the following primitive actions:

1. p-build-hard-top
2. p-install-sunroof
3. p-install-ac-compressor-4c
4. p-install-4c-pistons
5. p-build-4c-block
6. p-assemble-4c-engine
7. p-build-sedan-frame
8. p-build-body
9. p-paint-exterior
10. p-install-luggage-rack
11. p-install-engine
12. p-connect-ac
13. p-install-wiring

While this plan builds a car with a sedan frame, hard top, A/C, and sunroof, the car has a 4 cyl. engine instead of the target 8 cyl. turbo engine. In Step 2, the Plan Retriever probes memory with the goal 4 cyl. engine (plus features of the initial situation — e.g., location Detroit, etc.). As in Step 1, the probe would be generalized on failure. Here the probe succeeds, and the plans retrieved are ranked using the similarity metric. Here the highest ranking plan retrieved is plan Build-Engine-2 (a subplan of plan Build-Car-2 in Case-Build-Car-2), a plan for building an 8 cylinder fuel-injected engine with primitive actions:

1. p-build-8c-block
2. p-install-8c-pistons
3. p-install-turbo
4. p-assemble-8c-engine

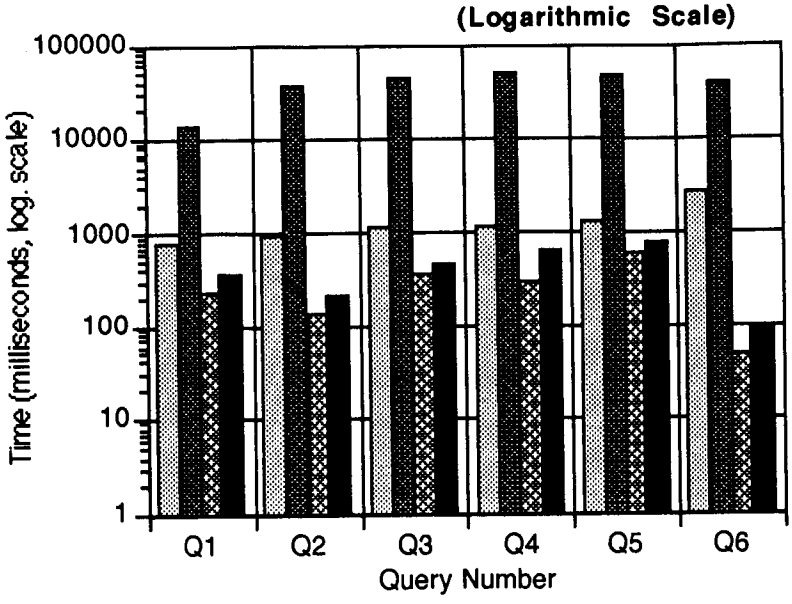
Using the concept hierarchy, CaPER judges the 8 cyl. fuel injected engine built by this plan to be closer to than the target goal of an 8 cyl. turbo engine than to a 4 cyl. engine, built by plan Build-Engine-1, a component plan of root source plan Build-Car-1. Build-Engine-1 is thus replaced by Build-Engine-2, which is predicted to require less adaptation to achieve the target goal.

For this sample target problem, two plans have been retrieved for adaptation. Plan Build-Car-1 built a car with a sedan frame, hard top, A/C, and sunroof. Plan Build-Engine-2 built an 8 cyl. fuel injected engine. The Plan Adapter will adapt these plans to the target situation and combine the new plans into a single plan for achieving all of the target goals.

4. EMPIRICAL RESULTS

To test the efficiency of CaPER's retrieval methods, several queries were issued via the PARKA Structure Retriever (described in section 2.1), which CaPER uses to probe its memory to retrieve cases and other episodes. The queries were done using both the serial and parallel versions for two casebases of 10 cases (1K PARKA frames) and 100 cases (8K frames). Each case was a plan for building a particular type of car that was generated by NONLIN for the simplified automotive assembly domain. The serial version of PARKA was run on a Macintosh IIcx. The parallel version was run on a CM-2 with 8K processors (16K virtual processors). The results are shown in figure 2.The queries, selected to be representative of those CaPER's Plan Retriever issues, were:

1. Find all plans for building a car in Detroit
2. Find all plans for building a car with a 4 cyl. engine
3. Find all plans for building a car with a coupe frame, convertible top, and a 4 cyl. engine
4. Find all plans for building a car with A/C
5. same as in (3) but also with a sunroof and A/C
6. find all plans for building any component in Detroit.



	Serial 10	Serial 100	Paral. 10	Paral. 100
1	772	13444	224	362
2	961	35389	133	217
3	1145	45155	367	469
4	1189	48223	297	637
5	1294	45278	603	758
6	2689	37522	47	92

Figure 2. Queries of CaPER Casebases Using PARKA Structure Retriever

These results show that serial retrieval methods clearly do not scale well to larger casebases. As can be seen, the runtime performance of the parallel algorithms is no worse than logarithmic in the size of the memory. It is expected that an optimized implementation of the structure retriever's parallel algorithms will produce a significant linear decrease in the retrieval times. The parallel system's runtime performance follows from the fact that retrieval times for PARKA queries specified without the structure retriever are dependent only on the depth of the memory network, not its size (Evett et al. 1992).

5. FUTURE WORK

The Plan Adapter component will adapt the source plan(s) found by the Plan Retriever to the target problem. We believe that less adaptation will be required than with traditional CBP systems since more specific plans can be retrieved. The adapted plans will then be merged into a new (composite) plan. Interactions between these subplans of the new plan need to be detected — either at adaptation time (possibly using mechanisms from generative planners) or during plan testing.

The Plan Failure Diagnoser and the plan repair mechanisms, making use of case-based techniques wherever possible, will repair faults (such as harmful interactions between subplans) in a new plan found by the Plan Tester. CaPER's memory will contain past failure episodes to be used to explain similar failures found in the current situation and past repair episodes to be used to fix similar problems.

Another goal is to test CaPER in more realistic domains. One characteristic of the simplistic automobile assembly domain is the modularity of plans to solve problems in it. Our car plans are modular since they assume the modular construction of cars: i.e., the subplan for building the engine is somewhat independent of the subplan for building the frame. This modularity allows subplans from different car plans to be more easily combined. The Plan Adapter has to check for fewer interactions between them, etc. It is therefore desirable to see how CaPER performs on "less modular" kinds of problems

One domain under consideration for implementation is a transport logistics domain involving deliveries of things by plane, truck, etc. This domain has problems that are less modular than those of the car assembly domain. Case retrieval in this domain would probably be highly sensitive to the initial situation of target problems (e.g., what

trucks are where, which cities have airports, etc.). The space of target problems is also very large. In contrast, problems in the car assembly domain have highly similar initial situations and a narrower range of goals.

6. RELATED WORK

Closely related to work done in CBR is work done in memory-based reasoning (MBR). MBRTalk is a MBR system for speech pronunciation that uses massive parallelism, as CaPER does, to access a large, unindexed memory of cases residing in the CM (Stanfill and Waltz 1986). MBRTalk retrieves cases that are the shortest distance from a probe. Each case, in parallel, computes its distance from the probe using a simple similarity metric. CaPER, in contrast, computes similarity scores for candidate cases in serial (due to cases being distributed across CM processors). MBRTalk uses a flat representation of cases. CaPER uses a highly structured, semantic network representation of cases. CaPER also uses a stronger domain model than MBRTalk.

PARADYME (Kolodner and Thau 1988) is an implementation of a memory for a CBR system that has much in common with parts of CaPER/PARKA. PARADYME uses the massive parallelism of the CM to access an unindexed memory. Memory holds concepts as well as cases. Cases are represented using a semantic net/frame-based representation. Cases are annotated with critical features (similar to CaPER's key features) to be used for selecting the best matching cases. PARADYME used different retrieval procedures than CaPER and was not yet integrated with a CBR system. Additionally, PARKA's retrieval algorithms are significantly more complete and faster than PARADYME's (Evett et al. 1992).

7. CONCLUSION

CBP systems can take advantage of plan re-use where possible. The success of this approach depends on the ability to retrieve old cases that are similar to the target problem and to adapt these cases appropriately. Using its unindexed memory and massively parallel retrieval mechanisms, CaPER can do fast, frequent retrievals using any features of the target problem. The result of this is that CaPER can retrieve cases that better match the target problem and thus require less adaptation. Empirical results indicate that the often inflexible, special-purpose indexing schemes required for the serial retrieval methods of traditional CBP systems using indexed memories can be abandoned in favor of parallel methods. These parallel methods can scale up to casebases of hundreds or thousands of cases. Larger casebases can improve the plans a CBP system produces by providing more cases to drawn upon.

REFERENCES

Gentner, D., "The mechanisms of analogical learning," In *Similarity and Analogical Reasoning*, Eds. S. Vosniadou and A. Ortony, Cambridge: Cambridge University Press, 1989, pp. 199-241.

Evett, M.P., Hendler, J.A., and Spector, L., "Parallel Knowledge Representation on the Connection Machine," *Journal of Parallel and Distributed Computing*, 1992 (forthcoming).

Hammond, K.J. (1990a), "Case-based planning: A framework for planning from experience," *Cognitive Science*, Vol. 14 (1990), pp. 384-443.

Hammond, K.J., (1990b), "Explaining and Repairing Plans That Fail," In *Artificial Intelligence*, Vol. 45 (1990), 173-228.

Hillis, W.D., *The Connection Machine*, Cambridge, Massachusetts: The MIT Press, 1985.

Kolodner, J.L., and Thau, R., *Design and Implementation of a Case Memory*, Technical Report RL88-1, Thinking Machines Corporation, Cambridge, Massachusetts, 1988.

Ross, B.H., "Some Psychological Results on Case-based Reasoning," In *Proceedings: Case-Based Reasoning Workshop (DARPA)*, San Mateo, California: Morgan Kaufmann, 1989, pp. 144-147.

Spector, L., Hendler, J.A., and Evett, M.P., *Knowledge Representation in PARKA*, Technical Report 2410, Department of Computer Science, University of Maryland at College Park, 1990.

Spector, L., Anderson, B., Hendler, J., Kettler, B., Swartzman, E., Woods, C., and Evett, M., *Knowledge Representation in PARKA - Part 2: Experiments, Analysis, and Enhancements*, Technical Report 2837, Department of Computer Science, University of Maryland at College Park, 1992.

Stanfill, C. and Waltz, D., "Toward Memory-Based Reasoning," *Communications of the ACM*, Vol. 29, No. 12, December 1986, pp. 1213-1228.

Tate, A., *Project Planning Using a Hierarchic Non-linear Planner*, Research Report No. 25, Department of Artificial Intelligence, University of Edinburgh.

Thagard, P.R. and Holyoak, K.J., "Why Indexing is the Wrong Way to Think About Analog Retrieval", In *Proceedings: Case-Based Reasoning Workshop (DARPA)*, San Mateo, California: Morgan Kaufmann, 1989, pp. 36-40.

Vosniadou, S. and Ortony, A., "Similarity and analogical reasoning: a synthesis," In *Similarity and Analogical Reasoning*, Cambridge: Cambridge University Press, 1989, pp. 1-17.

Waltz, D., "Is Indexing Used for Retrieval?" In *Proceedings: Case-Based Reasoning Workshop (DARPA)*, San Mateo, California: Morgan Kaufmann, 1989, pp. 41-45.

Agent Oriented Programming

an overview of the framework and summary of recent research

Yoav Shoham
Robotics Laboratory
Computer Science Department
Stanford University

This is a short overview of the *agent-oriented programming* (AOP) framework. AOP can be viewed as an specialization of *object-oriented programming*. The state of an agent consists of components called beliefs, choices, capabilities, commitments, and possibly others; for this reason the state of an agent is called its *mental state*. The mental state of agents is captured formally in an extension of standard epistemic logics: beside temporalizing the knowledge and belief operators, AOP introduces operators for commitment, choice and capability. Agents are controlled by *agent programs*, which include primitives for communicating with other agents. In the spirit of *speech-act theory*, each communication primitives is of a certain type: informing, requesting, offering, and so on. This document describes these features in a little more detail, and summarizes recent results and ongoing AOP-related work.

1 Introduction

Agent Oriented Programming is a proposed new programming paradigm, based on a societal view of computation. Although new, the proposal benefits from extensive previous research. Indeed, the discussion here touches on issues that are the subject of much current research in AI, issues which include the notion of agenthood and the relation between a machine and

its environment. Many of the ideas here intersect and interact with the ideas of others. In this overview, however, I will not place this work in the context of other work. That, as well as more details on AOP, appear in a Technical Report STAN-CS-1335-90 (revised), and subsequent publications which are mentioned below.

1.1 What is an agent?

The term 'agent' is used frequently these days. This is true in AI, but also outside it, for example in connection with data bases and manufacturing automation. Although increasingly popular, the term has been used in such diverse ways that it has become meaningless without reference to a particular notion of agenthood. Some notions are primarily intuitive, others quite formal. In several longer publications I outline several senses of agenthood that I have discerned in the AI literature. Given the limited space, here I will directly present "my" sense of agenthood.

I will use the term '(artificial) agents' to denote entities possessing formal versions of mental state, and in particular formal versions of beliefs, capabilities, choices, commitments, and possibly a few other mentalistic-sounding qualities. What will make any hardware or software component an agent is precisely the fact that one has chosen to analyze and control it in these mental terms.

The question of what an agent is now replaced by the question of what can be described in terms of knowledge, belief, commitment, *et cetera.* The answer is that *anything* can be so described, although it is not always advantageous to do so. D. Dennett proposes the "intentional stance," from which systems are ascribed mental qualities such as intentions and free will. The issue, according to Dennett, is not whether a system really is intentional, but whether we can coherently view it as such. Similar sentiments are expressed by J. McCarthy in his 'Ascribing Mental Qualities to Machines' paper, who also distinguishes between the 'legitimacy' of ascribing mental qualities to machines and its 'usefulness.' In other publications I illustrate the point through the light-switch example. It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmitting current at will, who invariably transmits current when it believes that we want it transmitted and not otherwise; flicking the switch is simply our way of communicating our desires. However, while this is a coherent view, it does not buy us anything, since we essentially understand

the mechanism sufficiently to have a simpler, mechanistic description of its behavior. In contrast, we do not have equally good knowledge of the operation of complex systems such as robots, people, and, arguably, operating systems. In these cases it is often most convenient to employ mental terminology; the application of the concept of 'knowledge' to distributed computation, discussed below, is an example of this convenience.

1.2 Agent- versus Object-Oriented Programming

Adopting the sense of agenthood just described, I have proposed a computational framework called *agent-oriented programming* (AOP). The name is not accidental, since from the engineering point of view AOP can be viewed as a specialization of the *object-oriented programming* (OOP) paradigm. I mean the latter in the spirit of Hewitt's original Actors formalism, rather than in some of the senses in which it used today. Intuitively, whereas OOP proposes viewing a computational system as made up of modules that are able to communicate with one another and that have individual ways of handling incoming messages, AOP specializes the framework by fixing the state (now called *mental state*) of the modules (now called *agents*) to consist of precisely-defined components called beliefs (including beliefs about the world, about themselves, and about one another), capabilities, choices, and possibly other similar notions. A computation consists of these agents informing, requesting, offering, accepting, rejecting, competing, and assisting one another. This idea is borrowed directly from the *speech act* literature. Speech-act theory categorizes speech, distinguishing between informing, requesting, offering and so on; each such type of communicative act involves different presuppositions and has different effects. Speech-act theory has been applied in AI, in natural language research as well as in plan recognition. To my knowledge, AOP and McCarthy's Elephant2000 language are the first attempts to base a programming language in part on speech acts. Figure 1 summarizes the relation between AOP and OOP.[1]

[1]There is one more dimension to the comparison, which I omitted from the table, and it regards inheritance. Inheritance among objects is today one of the main features of OOP, constituting an attractive abstraction mechanism. I have not discussed it since it is not essential to the idea of OOP, and even less so to the idea of AOP. Nevertheless a parallel can be drawn here too.

Framework:	OOP	AOP
Basic unit:	object	agent
Parameters defining state of basic unit:	unconstrained	beliefs, commitments, capabilities, choices, ...
Process of computation:	message passing and response methods	message passing and response methods
Types of message:	unconstrained	inform, request, offer, promise, decline, ...
Constraints on methods:	none	honesty, consistency, ...

Figure 1: OOP versus AOP

1.3 On the use of pseudo-mental terminology

The previous discussion referred to mentalistic notions such as belief and commitment. In order to understand the sense in which I intend these, it is instructive to consider the use of logics of knowledge and belief in AI and distributed computation. These logics, which were imported directly from analytic philosophy first to AI and then to other areas of computer science, describe the behavior of machines in terms of notions such as knowledge and belief. In computer science these mentalistic-sounding notions are actually given precise computational meanings, and are used not only to prove properties of distributed systems, but to program them as well. A typical rule in such a 'knowledge based' systems is "if processor A does not *know* that processor B has received its message, then processor A will not send the next message." AOP augments these logics with formal notions of choices, capabilities, commitments, and possibly others. A typical rule in the resulting systems will be "if agent A *believes* that agent B has *chosen* to do something harmful to agent A, then A will *request* that B change its choice." In addition, temporal information is included to anchor belief, choices and so on in particular points in time.

Here again we may benefit from some ideas in philosophy and linguistics. As in the case of knowledge, there exists work in exact philosophy on logics for choice and ability. Although they have not yet had an effect in AI comparable to that of logics of knowledge and belief, they may in the future.

Intentional terms such as knowledge and belief are used in a curious sense in the formal AI community. On the one hand, the definitions come nowhere close to capturing the full

linguistic meanings. On the other hand, the intuitions about these formal notions do indeed derive from the everyday, common sense meaning of the words. What is curious is that, despite the disparity, the everyday intuition has proven a good guide to employing the formal notions in some circumscribed applications. AOP aims to strike a similar balance between computational utility and common sense.

2 Overview of the AOP framework

A complete AOP system will include three primary components:

- A restricted formal language with clear syntax and semantics for describing mental state. The mental state will be defined uniquely by several modalities, such as belief and commitment.
- An interpreted programming language in which to program agents, with primitive commands such as `REQUEST` and `INFORM`. The semantics of the programming language will depend in part on the semantics of mental state.
- An 'agentifier,' converting neutral devices into programmable agents.

In the remainder of this document I will start with an short discussion of mental state. I will then present a general family of agent interpreters, a simple representative of which has already been implemented. I will end with a summary of recent research results and outstanding questions related to AOP.

3 On the mental state of agents

The first step in the enterprise is to define agents, that is, to define the various components of mental state and the interactions between them. There is not a unique 'correct' definition, and different applications can be expected to call for specific mental properties.[2]

[2]In this respect our motivation here deviates from that of philosophers. However, I believe there exist sufficient similarities to make the connection between AI and philosophy mutually beneficial.

In related past research by others in AI three modalities were explored: belief, desire and intention (giving rise to the pun on BDI agent architectures). Other similar notions, such as goals and plans, were also pressed into service. These are clearly important notions; they are also complex ones, however, and not necessary the most primitive ones. Cohen and Levesque, for example, propose to reduce the notion of *intention* to those of *goal* and *persistence*. We too start with quite basic building blocks, in fact much more basic that those mentioned so far. We currently incorporate two modalities in the mental state of agents: *belief* and *obligation* (or *commitment*). We also define *decision* (or *choice*) as an obligation to oneself. Finally, we include a third cateogry which is not a mental construct *per se*, *capability*. There is much to say on the formal definitions of these concepts; some of results described in the final section address this issue.

By restricting the components of mental state to these modalities I have in some informal sense excluded representation of motivation. Indeed, we do not assume that agents are 'rational' beyond assuming that their beliefs, obligations and capabilities are internally and mutually consistent. This stands in contrast to the other work mentioned above, which makes further assumptions about agents acting in their own best interests, and so on. Such stronger notions of rationality are obviously important, and I am convinced that in the future we will wish to add them. However, neither the concept of agenthood nor the utility of agent-oriented programming depend on them.

4 A generic agent interpreter

In the previous section I discussed the first component of the AOP framework, namely the definition of agents. I now turn to the central topic of this paper, the programming of agents, and will outline a generic agent interpreter.

The behavior of agents is governed by programs; each agent is controlled by his own, private program. Agent programs themselves are not logical entities, but their control and data structures refer to the mental state of the agent using the logical language.[3]

[3]However, an early design of agent programs by J. Akahani was entirely in the style of logic programming; in that framework program statements themselves were indeed logical sentences.

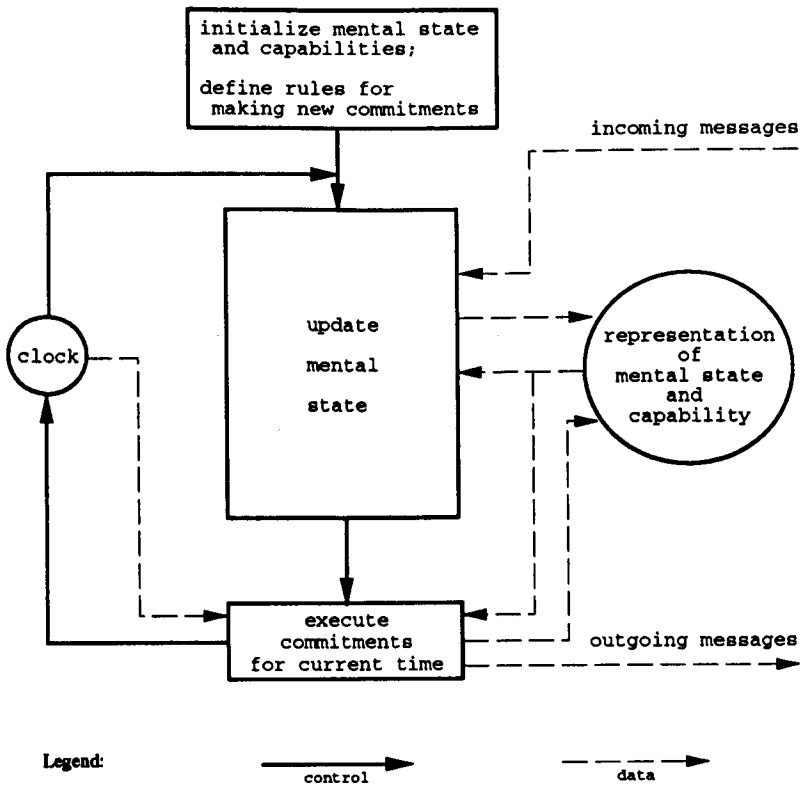


Figure 2: A flow diagram of a generic agent interpreter

The basic loop

The behavior of agents is, in principle, quite simple. Each agent iterates the following two steps at regular intervals:

1. Read the current messages, and update your mental state (including your beliefs and commitments);

2. Execute the commitments for the current time, possibly resulting in further belief change. Actions to which agents are committed include communicative ones such as informing and requesting.

The process is illustrated in Figure 2; dashed arrows represent flow of data, solid arrows temporal sequencing.

5 Summary of results and ongoing research

A more detailed discussion of AOP appears in [7]; the implemented interpreter is documented in [13]. Ongoing collaboration with the Hewlett Packard corporation is aimed at incorporating features of AOP in the New Wave™ architecture.

Preliminary ideas on the logic of mental state appear in [12]; a concrete proposal is made in [11]. This latter work addresses the properties of mental state at a given moment. Other publications address dynamic aspects of mental state. A logic for perfect memory and justified learning is discussed in [5]. [1] addresses the logic of belief revision; specifically, the postulates of belief update are shown to be derivable from a formal theory of action. The theory used there is the 'provably correct' theory presented in [3], which was later generalized to a frameowrk admitting concurrent action [4].

In parallel to the logical aspects of action and mental state, we have investigated algorithmic questions. We have proposed a specific mechanism for tracking how beliefs change over time, called *temporal belief maps* [2].

We are particularly interested in how multiple agents can function usefully in the presence of other agents. In [6] we propose the mechanism of *protograms* to balance conflicting influences of different agents. We are also interested in minimizing such conflicts in the first place, and have been investigating the computational utility of social law. In [10] we study the special case of traffic laws in a restricted robot environment; in [8] we propose a general framework for representing social laws within a theory of action, and investigate the computational complexity of automatically synthesizing useful social laws.

References

[1] A. del Val and Y. Shoham. Deriving the postulates of belief update from theories of action. Stanford working document, 1992.

[2] H. Isozaki and Y. Shoham. A mechanism for reasoning about time and belief. In *Proc. Conference on Fifth Generation Computer Systems*, Japan, 1992.

[3] F. Lin and Y. Shoham. Provably correct theories of action (preliminary report). In *Proc. NCAI*, Anaheim, CA, 1991.

[4] F. Lin and Y. Shoham. Concurrent actions in the situation calculus. In *Proc. NCAI*, San Jose, CA, 1992.

[5] F. Lin and Y. Shoham. On the persistence of knowledge and ignorance. Stanford working document, 1992.

[6] E. Mozes and Y. Shoham. Protograms. Stanford working document, 1991.

[7] Y. Shoham. Agent Oriented Programming. *The Artificial Intelligence Journal*, 1992 (to appear).

[8] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agents. In *Proc. NCAI*, San Jose, CA, 1992.

[9] Y. Moses and Y. Shoham. Belief as Defeasible Knowledge. *The Artificial Intelligence Journal*, 1992 (to appear; preliminary version appeared in IJCAI 89).

[10] Y. Shoham and M. Tennenholtz. On traffic laws for mobile robots. Stanford working document, 1992.

[11] B. Thomas. A logic for representing action, belief, capability, and intention, 1992. Stanford working document.

[12] B. Thomas, Y. Shoham, A. Schwartz, and S. Kraus. Preliminary thoughts on an agent description language. *International Journal of Intelligent Systems*, 6(5):497–508, August 1991.

[13] M. Torrance. The AGENT0 programming manual, 1991. Stanford technical report.

[14] Y. Shoham and M. Tennenholtz. Emergent Conventions in Multi-Agent Systems. In *Proc. KR*, Boston, MA, 1992.

DECISION THEORY FOR COMPUTING VARIABLE AND VALUE ORDERING DECISIONS FOR SCHEDULING PROBLEMS

Theodore A. Linden
Advanced Decision Systems
A Division of Booz, Allen & Hamilton
1500 Plymouth Street
Mountain View, CA 94043

Heuristics that guide search are critical when solving large planning and scheduling problems, but most variable and value ordering heuristics are sensitive to only one feature of the search state. One wants to combine evidence from all features of the search state into a subjective probability that a value choice is best, but there has been no solid semantics for merging evidence when it is conceived in these terms. Instead, variable and value ordering decisions should be viewed as problems in decision theory. This led to two key insights:

1) The fundamental concept that allows heuristic evidence to be merged is the net incremental utility that will be achieved by assigning a value to a variable. Probability distributions about net incremental utility can merge evidence from the utility function, binary constraints, resource constraints, and other problem features. The subjective probability that a value is the best choice is then derived from probability distributions about net incremental utility.

2) The methods used for rumor control in Bayesian Networks are the primary way to prevent cycling in the computation of probable net incremental utility.

These insights lead to semantically justifiable ways to compute heuristic variable and value ordering decisions that merge evidence from all available features of the search state.

Session A4: MONITORING AND CONTROL

Session Chair: Dr. Robert Lea

N93-32138

Implementing a Real Time Reasoning System for Robust Diagnosis

Tim Hill
William Morris
Charlie Robertson
McDonnell Douglas Space Systems Company
Space Station Division
16055 Space Center Boulevard
Houston, TX 77062
hill, morris, ccr
@kbs.mdc.com

Abstract

The objective of the Thermal Control System Automation Project (TCSAP) is to develop an advanced Fault Detection, Isolation, and Recovery (FDIR) capability for use on the Space Station Freedom (SSF) External Active Thermal Control System (EATCS). Real-time monitoring, control, and diagnosis of the EATCS will be performed with a Knowledge-Based System (KBS). This paper describes implementation issues for the current version of the KBS.

The TCSAP KBS is a combination of three distinct elements that interact with each other. The first is a quantitative model of the EATCS, providing step-wise steady state values for any EATCS configuration. The model is used in sensor validation and component diagnosis by comparing observed sensor readings with their computed values. Inconsistencies between observed and expected values imply either instrumentation failure or actual off-nominal behavior of the EATCS. The second element is a rule-based system containing safety critical and non-critical FDIR rules focused directly on the EATCS. The rules use both quantitative and qualitative values for reasoning and diagnosis. Quantitative sensor values are obtained from an external source and qualitative representations are derived from the history of the quantitative data. The third KBS element is the Human Interface (HI). The HI implements graphically oriented monitoring and control capabilities for the EATCS. The interface attempts to "intelligently" support the operator by supplying information of the type and quantity most likely needed in a given context. The HI also allows the user to specify configuration changes such as the closing or opening of a valve. These changes can be transmitted to the EATCS hardware as well as affecting the internal KBS quantitative model.

The KBS utilizes conventional software and a real-time expert system tool called G2. The use of G2 eases development of reasoning techniques required for automating the EATCS monitoring, control, and FDIR tasks. The resulting KBS utilizes a combination of model-based sensor validation, rule-based fault descriptions, and model-based diagnosis of unanticipated faults.

EATCS Overview

The External Active Thermal Control System (EATCS)[1] of Space Station Freedom (SSF) provides cooling and control necessary to maintain elements, systems, and components within their required temperature ranges. The EATCS design has evolved from the single-phase fluid system used in Apollo and Space Shuttle to a two-phase system (ammonia liquid and vapor mixture) on SSF. Both active and passive components of the EATCS can potentially fail or become blocked. As a result, a variety of failure modes exist. When this is combined with the continuous range of normal operating conditions and issues related to two-phase flow, EATCS diagnostics can become very complex.

The Space Station EATCS is a central facility, transporting waste heat away from crew quarters, experiment packages, computers, DC-to-DC power conversion units, etc., and radiating it into space. It utilizes ammonia as the working fluid and interfaces via heat acquisition devices (HADs) with the habitation and laboratory modules, and truss mounted equipment where the heat dissipation rates are too high to be controlled passively. HADs are heat exchangers and cold plates that remove heat directly from fluid systems and electronic equipment. Liquid ammonia is supplied to the HADs by the EATCS and is vaporized by the particular heat load being serviced. The vapor is transported to the radiators which reject heat to space. Figure 1 shows a functional configuration of the major EATCS components included in a single loop or bus.

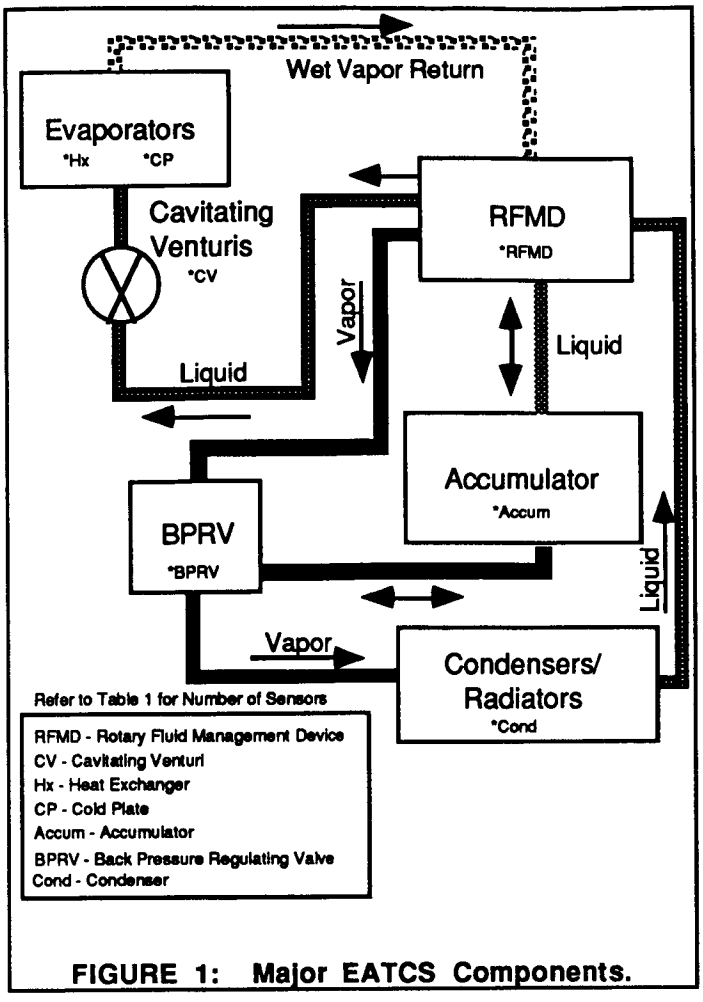


FIGURE 1: Major EATCS Components.

Table 1 shows the number of sensors for the major components shown in Figure 1.

TABLE 1: Number of Sensors and Location

Loc.	Temp	Press	DP	Flow	Other
RFMD	5	2	3	2	2
CV (5)	5	0	5	5	0
Hx (3)	6	6	0	0	0
CP (2)	2	2	2	0	0
Accum (2)	2	0	1	0	2
BPRV	1	0	0	0	1
Cond (2)	6	5	3	2	0
TOTALS	27	15	14	9	5

KBS Overview

The Thermal Control System Automation Project (TCSAP) Knowledge-Based System (KBS) is a combination of three distinct elements that interact with each other.[2,3] The first is a quantitative model of the EATCS, providing step-wise steady state values for any EATCS configuration. The second element is a Rule-Based System (RBS) containing safety critical and non-critical FDIR rules focused directly on the EATCS. The third KBS element is the Human Interface (HI). The KBS utilizes conventional software and a real-time expert system tool called G2.

Model Development and Use

The internal simulation model used by the KBS for sensor validation is an object-oriented reconfigurable model centered around the actual major EATCS components and their connectivity (Figure 2). The model is used in three key areas by the KBS as discussed further in this paper: (1) to perform sensor validation, (2) to provide transition points for mapping sensor data to qualitative states in the RBS, and (3) to provide expected operating conditions to the HI. Each component modeled contains state variables associated with inlet and outlet conditions. These state variables are flow rate, temperature, pressure, and quality (vapor mass divided by the sum of vapor and liquid mass). Constraints are represented by generic rules and mathematical formulae that govern the relationships among these state variables and their propagation through the components of the thermal bus. These constraints are based on the laws of physics and thermodynamics (e.g. conservation of mass), as well as the actual component design parameters (e.g. device specific pump head curves).

Sensor objects can be logically connected at the inlet or outlet of any component in the model to represent actual sensor locations. At these points, sensor readings can be compared with the model-predicted value to perform sensor validation and initiate component fault diagnosis. Reconfiguration of the model occurs automatically from changes in heat loads, pump speed, set point temperature, or isolation valve positions in the monitored hardware.

Several simplifying assumptions were made during the building of the model. The foremost simplification in the current model is to support only steady state conditions. The model "propagates" from one steady state to the next without regard to time lags or transient states. Another significant assumption is the adiabatic behavior (no heat transfer) of all components except heat exchangers. Other assumptions in the current model are that the bus will absorb the total heat load, and that flow will always be in one direction.

A major goal in building this model was that it be as generic as possible, so that maintenance would be greatly simplified. Another goal was to have a model which could support model-based reasoning for sensor validation and component diagnosis. We do not attempt to encode a global set of equations or a solution strategy in this model. Instead, state variables are propagated across components and from one component to another either upstream or downstream according to the generic constraints previously discussed. This propagation occurs until all state variables converge to steady values (within some tolerance).

Input design and configuration parameters for the model include the flow-rate of every cavitating venturi, the design heat load of every evaporator, the design heat rejection capability for each condenser, the initial accumulator positions, and all valve states. The total condenser cooling capacity and the estimated average condenser subcooling are also calculated. Regularly supplied external configuration information includes: the heat load on each evaporator, the commanded setpoint temperature, and the RFMD motor speed. Using these facts and assumptions the KBS model is able to propagate between steady states.

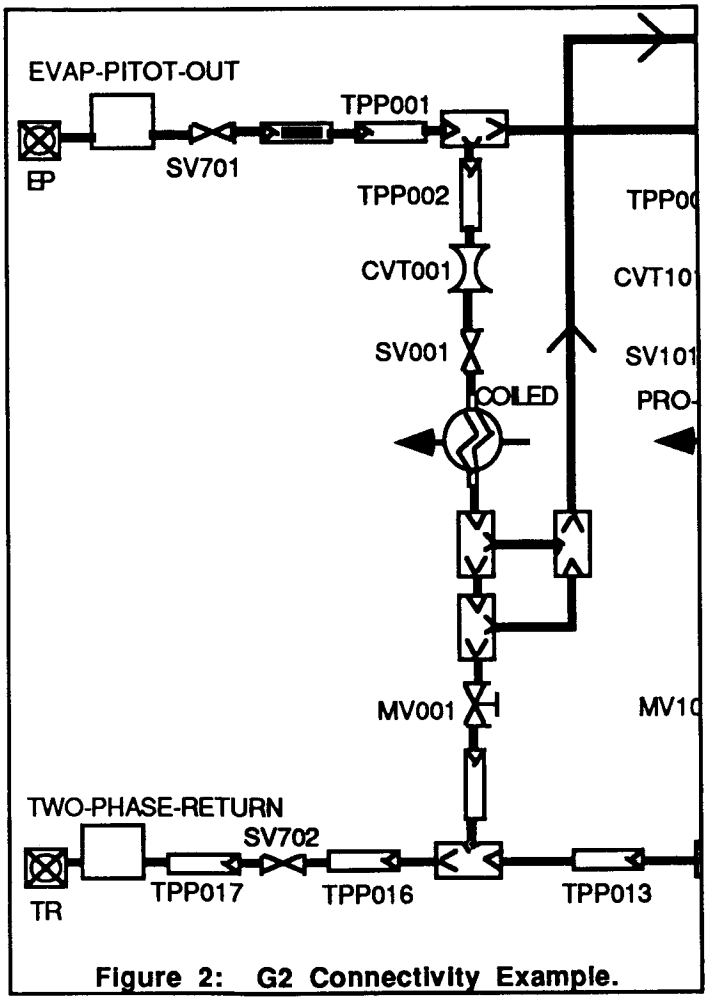


Figure 2: G2 Connectivity Example.

The KBS model is propagated with the assumption that the bus is operating nominally. In fact, both nominal and off-nominal operating conditions cover continuous ranges. For example, if one evaporator has been shut down, the re-configured bus can still be viewed as operating nominally. The bus would reach a new steady state reflecting a configuration with one less evaporator. Similarly, there is no discrete distinction between off-nominal behaviors. Conditions considered nominal in one state might represent a fault scenario in a different operational mode. An infinite number of combinations are possible. This makes the use of a model even more appropriate in attempting sensor validation and fault diagnosis.

For sensor validation, the observed sensor readings at each point are compared with the corresponding computed values in the model. If the two values (sensor reading versus computed value) are not within tolerance then the sensor is marked as suspect. If sensor validation finds a sensor reading to be suspect, then the sensor will be monitored for fifteen seconds. Fifteen seconds is ample time to allow other sensors to go out of their application limit range, indicating that a fault is occurring. After fifteen seconds, if the sensor reading has not changed or the model still invalidates it, then that sensor is automatically failed. A failed sensor on-orbit may not be replaceable for several months, so working with the instrumentation available is imperative. Each sensor definition has the capability of specifying backups. A more detailed description of the backup sensor implementation is given in the next section (Rule-Based System).

The model-based reasoning for component diagnosis portion of the KBS is still in development. Several different approaches [4,5,6,7] are being implemented in parallel efforts. The strengths and weaknesses of each technique will be described in a separate report.

Rule-Based System

The design intention of the KBS is to represent thermal expertise in the same way it is expressed by the thermal engineer. The comparison of observed values to expected values as described earlier is exactly what a human expert does implicitly. Complimenting this model-based view of the problem an expert also applies heuristics, lessons learned from experience, that can often be expressed in the form of if-then rules. The KBS uses such forward chaining rules in a Rule-Based System (RBS) to perform fault diagnosis.

The TCSAP RBS attempts to match rule conditions against patterns of system status information. The RBS has several advantages over traditional table-driven approaches to diagnosis which also match sensor readings to target values (in tables). The RBS rules represent diagnostic knowledge at a high level, allowing easier human interpretation, maintenance, and meaningful explanation capabilities. The RBS is not tied to a specific EATCS configuration. The data-driven nature of rule-based systems combined with support code for primary and alternate instrumentation allows the RBS to degrade more gracefully than a table lookup approach.

The FDIR rules reference qualitative states (e.g. low, nominal, high) and trends (e.g. decreasing, steady, increasing) to aid in development, interpretation, and maintenance of the RBS. Each sensor has qualitative mappings for current value and trend information[8]. Transition points for qualitative states are predefined but

can be dynamically modified such that a quantitative value range corresponding to a qualitative state of "low" may be different depending on the current system mode. For example (see Table 2), the qualitative state of a particular thermocouple might be nominal if the reading is above 58°F and below 66°F with a setpoint temperature of 62°F. Low, very-low, high, and very-high would map to other ranges. The trend of each sensor is translated in a similar way. The qualitative trend states include: rapidly-decreasing, decreasing, steady, increasing, and rapidly-increasing.

TABLE 2. Quantitative to Qualitative Mapping.

State	Transition Points
VERY-HIGH	
	70°F
HIGH	
	66°F
NOMINAL	
	58°F
LOW	
	54°F
VERY-LOW	

The rules are not tied to a specific configuration or load status of the EATCS. System state changes due to heat load variation, valve manipulation, or setpoint change may all be part of a normal operating plan. Since the RBS is integrated with the EATCS model described earlier, expected/computed values for a new thermal bus state are used to dynamically change transition point ranges. These new ranges then map incoming sensor data to an appropriate qualitative state. Figure 3 shows a simplified FDIR rule and some of the types of qualitative terminology used.

```
if rfmd-motor-speed is rapidly-decreasing and
    rfmd-end-to-end-deltap is low and
...
then
    ... and
    send-alarm("Evidence of an RFMD motor
failure", medium-priority)
```

Figure 3: The FDIR rules reference qualitative states and trends.

Table-driven systems and traditional rule-based implementations tend to degrade rapidly in the presence of failed sensors. In the TCSAP RBS, if a sensor has been failed by the model-based sensor validation routine the system attempts to automatically use backups. If a sensor that has been failed has a backup, then the backup sensors' reading will be provided for the failed sensors' reading. This allows the FDIR rules to function without change even if the primary sensors they refer to have failed. A backup sensor can be an actual sensor or a calculated value (e.g. a delta pressure calculated by two existing and "good" pressure sensors). If an actual sensor exists that can be used as a backup sensor for another, then the actual sensor is preferred over a calculated value.

Human Interface

The Human Interface (HI) allows the operator to monitor the status of the EATCS hardware and to understand the reasoning behind KBS messages and activities. G2 allows the HI to be built interactively on windows, called workspaces. Note that multiple workspaces may exist on the screen at a time. A workspace called the Schematic Browser allows quick and easy access to different contexts and varying levels of detail. Figure 4 shows the check-box format of the Schematic Browser workspace.

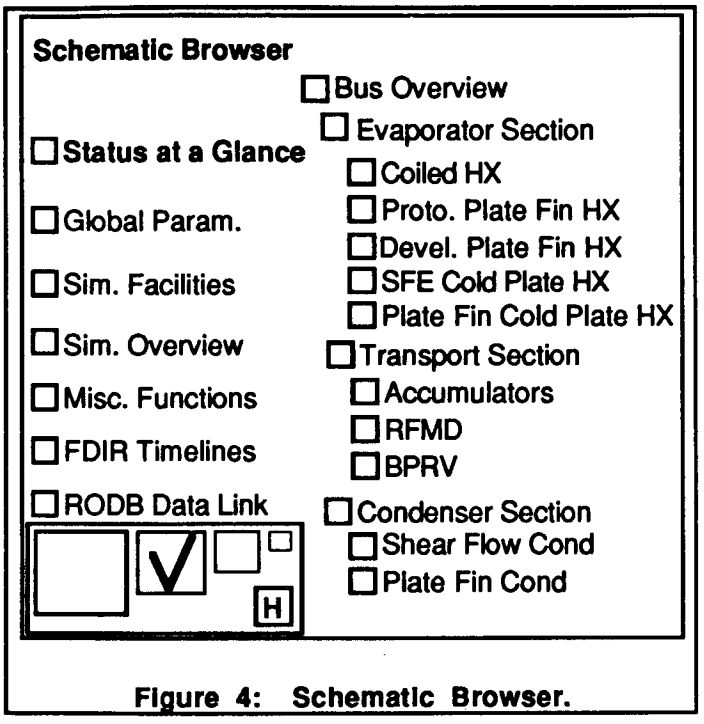


Figure 4: Schematic Browser.

The Status-at-a-Glance workspace (Figure 5) was developed to show the relationships between key values and is generally the most useful for monitoring purposes[9]. Since the EATCS is designed to maintain a constant heat sink temperature for station heat loads, the evaporator liquid supply temperature is a crucial measure of system performance and status. At the top of the screen are the Evaporator and Setpoint/System temperatures. At the bottom of the screen the exit quality and subcooling are used to present high-level evaporator and condenser loop status. The Mass Gauging and RFMD displays in the center show the status of transporting liquid and vapor throughout the system. By normalizing the observations with their expected values, a high or low sensor reading is immediately visible as an extension of its bar chart above or below the normalization (horizontal) line.

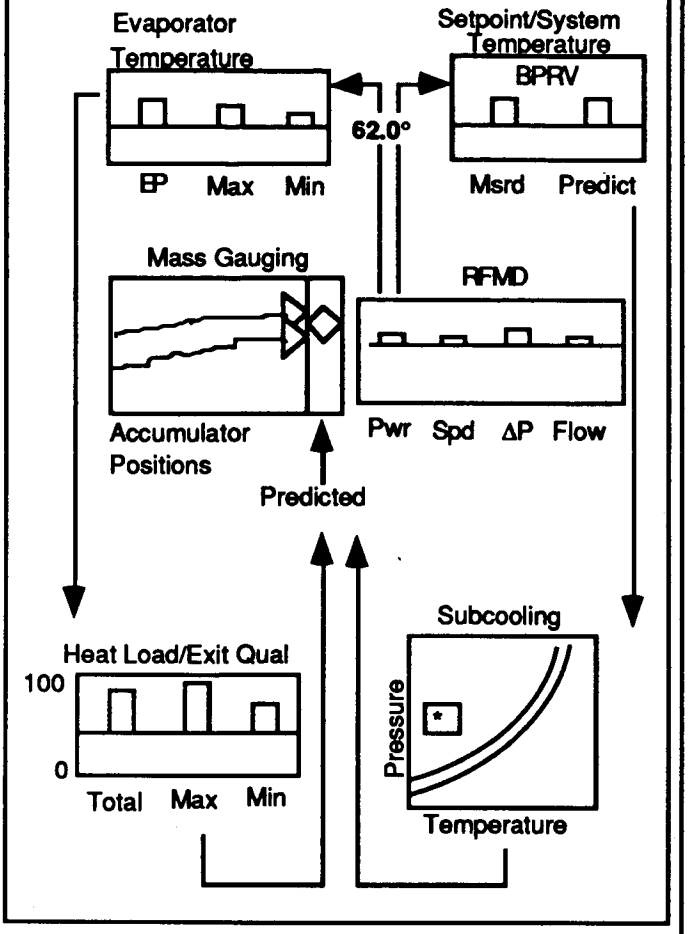


Figure 5: Status at a Glance.

The FDIR Timeline workspace notifies the operator of sensor validation and diagnostic messages (Figure 6). By clearly distinguishing between sensor validation messages, low priority fault messages, medium priority fault messages, and high priority fault messages, the Timeline gives the operator several tools for handling crucial messages and delaying action on lower priority information. In a real-time situation several messages can scroll off the Alarm workspace before the operator has a chance to respond. Each time a message is sent from the KBS to the Human Interface, the corresponding Timeline shows a "Blip". Any messages that might have scrolled off of the Alarm workspace too quickly are still visible as blips on the Timeline. The blips visually represent time-relative placement of each message to the other messages. By selecting the icon beside the desired message timeline, an operator can display messages of that specific priority on a separate workspace. Selection of individual messages allows access to more specific information about the diagnosis. The WHY option on a message displays time histories of sensors, pseudos, and simulated values pertinent to the diagnosis.

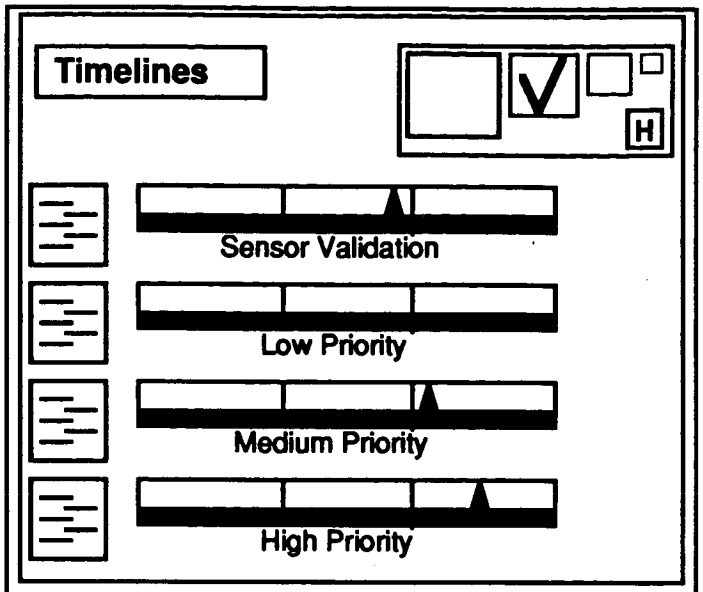


Figure 6: FDIR Timeline.

The second column on the Schematic Browser references workspaces that present a more detailed placement of instrumentation than the Status-at-a-Glance screen. These can be displayed individually as the Evaporator, Transport, and Condenser sections of the bus. They can also be displayed simultaneously using the Bus Overview selection.

The component schematics show the highest level of detail available through the HI. These screens present individual evaporators, condensers, accumulators, the BPRV, the RFMD, and all available instrumentation for each.

The Simulation Facilities and Simulation Overview screens provide a view of the internal KBS model readings. These show exactly where the KBS expects the hardware to be and allow the operator to manually adjust key parameters of the internal simulation.

A sample sequence of actions might begin with a flight controller observing an increase in total quality on the Status-at-a-Glance screen. The controller then displays the Evaporator Section to determine if any or all of the evaporator outlet temperatures are high. Subsequent actions might take the controller directly down to a detailed evaporator schematic or over to view the Transport Section. As the anomaly continues, the KBS issues messages validating the sensors and warning of "Evaporator Blockage" on a single evaporator. This activates an Alarms workspace and brings it to the top of the screen. By now, the controller may have gone to the Simulation Overview screen to compare observed evaporator loop conditions with calculated predictions from the model. Alternatively, the operator might "click" on the warning message and request an explanation of "WHY" the KBS made this diagnosis. In this example, plots of evaporator inlet and outlet temperatures, flow,

and delta-pressures would be presented. Several combinations of these readings could be indicative of some type of blockage.

Summary

Using a combination of conventional programming, rule-based technology, and model-based reasoning, the KBS is able to monitor, control, and perform FDIR on the SSF EATCS. Using an internal simulation model, the KBS can perform sensor validation and component diagnosis by comparing observed sensor readings with their computed values. The qualitative representations mapped by the model and used in the rule-based portion of the system increase flexibility and robustness. The KBS human interface makes use of data in the internal model to focus the controller on areas of inconsistent behavior.

Acknowledgements

The authors would like to acknowledge Roger Boyer for his contributions to the KBS development. Roger has provided invaluable thermal expertise and suggestions.

References

1. R.L. Boyer, "Space Station Freedom External Active Thermal Control System High Fidelity Simulation Modeling Document", McDonnell Douglas Space Systems Company, 1992.

2. W. Morris, T. Hill, C. Robertson. "Advanced Fault Management for the Space Station External Active Thermal Control System", SAE 22nd International Conference on Environmental Systems, Technical Paper Series, July 1992.

3. T. Hill, W. Morris, R. Boyer, "Thermal Control System Automaton Project (TCSAP) Interim Report", JSC 25448, MDC 91H01242, December 1991.

4. J. Collins, K. Forbus, "Building Qualitative Models of Thermodynamic Processes", report on work funded in part by NASA.

5. Y. Xudong, G. Biswas, "A Multi-level Diagnosis Methodology for Complex Systems", to appear in Proceedings IEEE CAIA-92.

6. J. Sticklen, A Kamel, W. Bond, "Integrating Quantitative and Qualitative Computations in a Functional Framework", Engineering Applications of Artificial Intelligence, Vol 4, No 1, pp 1-10, 1991.

7. C. Robertson, "PD12-503 Robust Fault Diagnosis and Management", McDonnell Douglas Space Systems Company IRAD Report, December, 1991.

8. B. Glass, Erickson and Swanson, "TEXSYS: A Large Demonstration of Model-Based Real Time Control of a Space Station Subsystem", 1991.

9. S. Potter, et al., "Visualization of Dynamic Processes: Function-Based Displays for Human-Intelligent System Interaction", to appear in Proceedings of the 1992 IEEE International Conference on Systems, Man, and Cybernetics, October 1992.

REINFORCEMENT LEARNING BASED ROBOTIC ARM CONTROL

Dr. Hamid Berenji, Dr. Robert Lea, Dr. Yashvant Jani, and Jeff Hoblitt
NASA/Johnson Space Center
2101 Nasa Road 1
Houston, TX 77058

Abstract unavailable at time of publication.

Attention Focussing and Anomaly Detection in Real-time Systems Monitoring

Richard J. Doyle, Steve A. Chien, Usama M. Fayyad, and Harry J. Porta
Artificial Intelligence Group
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099

Abstract

In real-time monitoring situations, more information is not necessarily better. When faced with complex emergency situations, operators can experience information overload and a compromising of their ability to react quickly and correctly. We describe an approach to focusing operator attention in real-time systems monitoring based on a set of empirical and model-based measures for determining the relative importance of sensor data.

Introduction: Sensor Selection

Mission Operations personnel within NASA are beginning to face the manifestations of a technology race. Our ability to devise safe, reliable monitoring strategies is not keeping pace with our ability to build space platforms of increasingly complex behavior with large numbers of sensors. To date, spacecraft such as Voyager have had sensor complements numbering only in the hundreds. For these space platforms, it has proven both feasible and appropriate to adopt a comprehensive monitoring strategy where mission operators interpret all of the sensor data all of the time.

However, NASA is moving into an era where sensors on space platforms such as Space Station Freedom will be numbered in the thousands. With space platforms of this complexity, the comprehensive monitoring strategy will be no longer tenable. This trend is not unique to NASA.

It is our thesis that for complex systems with large sensor complements a selective monitoring strategy must be substituted for the comprehensive strategy. The subject of our work is an approach to determining from moment to moment which subset of the available sensor data for a system is most informative about the state of the system and about interactions occurring within the system. We term this process *sensor selection* and we have implemented a prototype selective monitoring system called SELMON [Doyle and Fayyad 91, Chien et al 92, Doyle et al 92].

The SELMON system has its origins in a sensor planning system called GRIPE [Doyle et al 86] which planned information gathering activities to verify the execution of robot task plans. The goal of the current SELMON project is to provide assistance to operators by focusing their attention during real-time monitoring. Our sensor selection approach also could be embedded as part of an autonomous monitoring and control system.

Approach: Sensor Ordering

Our approach to focusing operator attention in real-time monitoring involves defining a set of sensor scoring measures. Each of these measures embodies a different viewpoint on why, at a particular moment, one sensor may be more worthy of operator attention than others. The measures are based in concepts from model-based reasoning and information theory. Some of the measures utilize sensor value predictions generated by simulating a causal model of the system being monitored.

During each timestep all sensors are scored according to these measures. The scores are used as a basis for an ordering on the sensors. See Figure 1. These scoring measures are divided into two categories. The first set – empirical methods – rely upon current and historical data to determine importance. These measures include *surprise*, *alarm*, *anticipate alarm*, and *value change*. The second set uses a causal model of the system to reason about expected current and future performance of the system to determine sensor importance. These methods include *deviation*, *sensitivity*, and *cascading alarms*.

After describing each of these measures, we describe how these measures are combined into an overall importance score for each sensor.

Empirical Sensor Scoring

In this section, we describe the empirical measures that are used in determining the overall importance score assigned to each sensor. This part of the score is based on four measures: *surprise*, *alarm*, *anticipate alarm*, and *value change*. These measures use knowledge about each individual sensor, independently of any knowledge about the interconnectedness of the sensors.

Surprise In order to obtain an ordering on the set of sensors, we need to quantify the following notions: How reliable is a sensor? How stable is it? How often does it go into an alarm state?

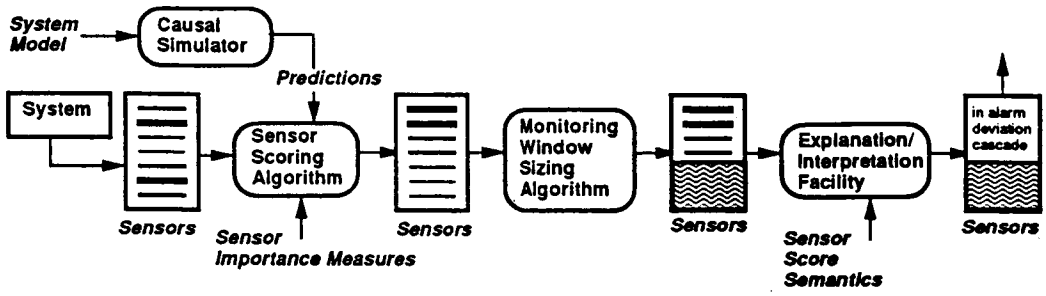


Figure 1: SELMON Architecture.

From an information theoretic point of view, a change in the value of a sensor gives us a certain amount of information (usually measured in bits). Assume we have two sensors, S_A and S_B. Further assume that sensor S_A's value has been wildly changing over the last 100 readings, while sensor S_B's value has been constant. If we are told that according to the latest update, the values of both sensors have increased by 25%, which do we consider a more informative event? Clearly the fact that S_B's value changed is more informative since it is more unusual. Prior to the latest reading, if we were asked to predict the values of S_A and S_B, then based on previous data, we would naturally guess that S_A's value is likely to have changed while S_B's value is likely to have remained constant. Then the fact that S_B changed value tells us something that we did not know or expect.

For each sensor, a cumulative histogram of its values is maintained for each system operating mode. This is done by dividing its range into a fixed number of bins. The boundaries between bins are determined through specific knowledge of the sensor and of the "interesting" subranges in its range. This histogram is then used to determine two measures of the interestingness of the most recent value returned by a sensor.

Denote the range of sensor S by $Range(S)$. If S is a continuously valued sensor, we can discretize its range into a set of collectively exhaustive ranges $\{R_1(S), R_2(S), \ldots, R_K(S)\}$, where

$$Range(S) = \bigcup_{i=1}^{K} R_i(S)$$

With each range $R_i(S)$ we associate a frequency measure $f_i(S)$ that gives the proportion of time that S's value has been in this range. Thus $f_i(S)$ is an estimate of the probability of the value of S falling in range $R_i(S)$ and

$$\sum_{i=1}^{K(S)} f_i(S) = 1$$

To quantify the degree to which sensor S is stable in its reading, we apply the notion of information entropy. The entropy of the values of a sensor S, denoted by $VEntropy(S)$, is defined by

$$VEntropy(S) = -\sum_{i=1}^{K} f_i(S) \cdot \log f_i(S)$$

where $VEntropy(S)$ is maximum when all ranges of values of S are equally likely (i.e., when S changes value often). It is minimum when the values of S have all been in one range $R_i(S)$, thus $f_i(S) = 1$ (for some i, $1 \leq i \leq K(S)$). It can easily be shown that $0 \leq VEntropy(S) \leq \log K$. We are now ready to define the average value informativeness of sensor S, denoted by $VInform(S)$, to be

$$VInform(S) = 1 - \frac{VEntropy(S)}{\log K(S)}$$

where $VInform(S)$ takes on values between 0 and 1. A value of 1 indicates that S normally rarely changes its value, while a value of 0 indicates that S's value is equally likely to be in any of its ranges.

On the other hand, the quantity

$$VUnusual(S) = 1 - f_i(S)$$

gives the unusualness of sensor S's value being in the i-th bin. $VUnusual(S)$ is computed each time S reports a value, and the i used is the index of the bin containing the reported value. This measure can assign the same degree of unusualness in fundamentally different situations. For instance, it does not distinguish between a value having a probability of $\frac{1}{K}$ occurring when all other values have an equal probability of $\frac{1}{K}$ each, and a value with probability $\frac{1}{K}$ when only one other value has probability $(1 - \frac{1}{K})$ with the remaining values having probability 0. In the first case, the value is just as likely as any other. In the second case, the interesting event is that the most likely value did not occur. To make this distinction we combine the unusualness and value entropy measures to obtain the *surprise* score:

$$Surprise(S) = VInform(S) \cdot VUnusual(S).$$

This measure takes on the maximum value of 1 when one bin in the histogram has probability one and the sensor registers a value in another bin. It has a minimum value of zero when all bins in the histogram are equally likely.

Accounting for Alarm Thresholds Alarm thresholds for sensors, indexed by operating mode, typically are established through an offline analysis of the design of NASA space systems. SELMON makes use of alarm threshold information in the following way: A sensor whose value traverses the safety threshold is said to go into a state of alarm. The predicate $In\_Alarm(S)$ captures this notion:

$$In\_Alarm(S) = \begin{cases} 1 & \text{if } S \text{ is outside its safety range} \\ 0 & \text{if } S \text{ is within its safety range} \end{cases}$$

We compute the value of an alarm score for S as follows:

$$ALScore(S) = In\_Alarm(S) \cdot [1 + Trav(S)] .$$

where $Trav(S)$ is the proportion of the alarm range traversed.

We consider alarms as interesting events whose importance decreases with time. Thus a sensor that persists in alarm state for prolonged periods of time should gradually fade from our attention. To achieve this we add an exponential decay factor. Let $t_A(S)$ be the time at which sensor S last entered into alarm. At any time t, the alarm score is computed as follows:

$$Alarm\_Score(S) = \frac{1}{2} ALScore(S) e^{-\beta(t - t_A(S))}$$

where $\beta > 0$ is the time decay constant. β is chosen small so the decay will not be too fast; typically $\beta \leq 0.1$/second.

Given the recent values of S, one may conduct a simple form of trend analysis to decide whether or not sensor S is anticipated to be in alarm soon. The measure $Predict\_Alarm(S)$ is a curve-fitting prediction of when the sensor will enter alarm. This measure has a minimum of 1 and a maximum of infinity if the curve fit indicates that the sensor will never enter alarm. If the sensor is currently in alarm, $Predict\_Alarm(S)$ measures when the sensor is predicted to leave alarm. This measure is used to compute a score *Anticipate Alarm* as follows:

$$Anticipate\_Alarm(S) = \begin{cases} 1/Predict\_Alarm \\ 1 - 1/Predict\_Alarm \end{cases}$$

The first case applies when S is within its safety range. The second case applies when S is outside its safety range.

Thus, if S is currently not in alarm, *Anticipate_Alarm* will be at its maximum of 1 when *Predict_Alarm* predicts the sensor will enter an alarm range immediately. If S is currently not in alarm, *Anticipate_Alarm* will be at its minimum of 0 when *Predict_Alarm* predicts the sensor will never enter alarm. If S is currently in alarm , *Anticipate_Alarm* will be at its maximum of 1 when *Predict_Alarm* predicts the sensor will never leave the alarm range. If S is currently in alarm, *Anticipate_Alarm* will be at its minimum of 0 when *Predict_Alarm* predicts the sensor will immediately leave alarm.

Quantifying Value Change A change in the value of a sensor is considered to be an event of interest. The surprise measure described above measures the degree of interestingness of a sensor taking on a certain value. Another aspect of sensor behavior to measure is the most recent change in value of the sensor that brought it to its current reading. However, absolute change magnitude is not interesting in and of itself. What is interesting is the probability of the most recent change taking place. Hence we need a scheme for normalizing the absolute change in value of a sensor.

The scheme we use assigns a score to each change in the value of a sensor that is an estimate of the proportion of all previous value changes for that sensor that had value changes strictly less than the change under consideration. Suppose we get a change in value of the sensor equal to Δ. Furthermore, suppose that 60% of the previous value changes for this sensor in the current operating mode have been less than Δ. In this case, we assign a score of 0.6 to the change Δ. Changes with magnitude greater than Δ will get higher scores.

This scheme requires that we keep track of a sorted sequence of all value changes of each sensor. This is neither feasible nor necessary. An approximation of this value can be obtained by keeping a constant number of values, say W, in a sorted sequence. Let the total number of changes in the values of a sensor so far be $C(S)$. Rather than storing all $C(S)$ values, we store only $W < C(S)$ values. With the arrival of a new change in value for sensor S, we increment the count of changes $C(S)$ and then we decide whether to replace one of the W values we are storing or simply ignore the current value change. The decision criterion is to generate a random number in $[0, 1]$ according to a uniform distribution, and replace one of the W values if and only if that random number is less than $\frac{W}{C(S)}$. It can be proven that this algorithm is equivalent to one that stores all $C(S)$ values, randomly samples W of them, and returns as score the proportion of the W elements that have value less than the change under consideration.

We call this score the *percentile value change* score. It is used to assign a normalized score in the range $[0, 1]$ for each change of value that occurs in each sensor. By definition, this score is maximum when the change is the maximum change of value seen so far for a particular sensor. It is minimum when no change occurs in the value of a sensor.

Model-Based Measures

SELMON also uses a model of the monitored system to determine sensor importance. This model is used to compute three scores: deviation, sensitivity, and cascading alarms. This section describes how each of these scores is computed.

Deviation The deviation measure uses a model of the monitored system to make predictions of expected current sensor readings. The concept of the deviation score is that sensor readings deviating significantly from the

predicted values are anomalous and should be reported to the operator.

The deviation score is computed in the following manner. First, the raw deviation is computed as the difference between the predicted and observed sensor scores. This raw deviation is entered into a normalization process identical to that used for the value change score, and the resultant score in the range [0,1] is the overall deviation score.

Causal Analysis The SELMON system also uses the causal model of the monitored system to reason about future effects of current quantity changes. These future effects are considered in two causal-based measures. First, *sensitivity* measures the effect of predicted changes in quantities on the overall state of the system. This is done by projecting each predicted change in a quantity individually forward as a perturbation of the system, and measuring the overall change in the system. Those currently occurring changes which have a greater effect upon the future state of the system are likely to be more important and thus receive high scores to be displayed to the operators. The second causal reasoning measure is *cascading alarms*, which measures the potential for observed changes to result in rapidly developing alarm sequences. The cascading alarms measure uses the same perturbation analysis used in the sensitivity analysis and measures the number of alarms triggered and how quickly alarms occur. Those predicted changes which are expected to trigger large numbers of alarms are scored highly and thus will be selected to be displayed to operators.

Sensitivity Analysis Sensitivity analysis measures the sensitivity of other quantities in the monitored system to changes in each quantity in the model. This is performed as follows. Beginning with a simulation of the system in its current state and time $T_{current}$, simulate forward one timestep (i.e. until the next time sensors are expected to be polled). For each quantity Q, choose ΔQ_{pred} as the current 50th percentile value change recorded for the given sensor.

Then, for each quantity Q, run a simulation beginning again with the current system state, perturbing Q by ΔQ_{pred}, propagating this change to other quantities in *All_Quantities* (the set of all quantities in the model) as dictated by the model. For each such changed quantity Q' in *All_Quantities*, for each time *time*1 that the quantity changes during the simulation, collect a sensitivity score proportional to the amount of change in Q' normalized to the size of the nominal range of the sensor but also modified by a decreasing function of *time*1. This calculation captures the characteristic that delayed and less direct effects are more likely to be controllable and less likely to occur. Thus, a change which affected a quantity Q' but occurred slowly is considered less important. This simulation proceeds for a predetermined amount of simulated time. Then, for each changed quantity Q', take the maximum of the collected *change_scores* for that quantity. The sensitivity score for Q is the sum of these maximums for all the Q's. Thus, for each quantity Q, a simulated change produces a set of *change_scores* for each other quantity in the model. The sensitivity score for Q is the sum of the respective maximums of each of these sets. If there are no changes to a quantity, this set is empty and the quantity receives a zero score.

A background sensitivity score is subtracted from the sensitivity score for Q, computed by measuring the sensitivity score via simulation with no perturbation of the system.

Cascading Alarms Analysis Cascading alarms analysis measures the potential for change in a single quantity to cause a large number of alarm states to occur, thus causing information overload and confusion for operators. In the cascading alarms score, the same simulation used in the sensitivity score computation is used to also determine the number of alarms triggered by the observed change. In the cascading alarms score, for each quantity Q, the number of alarms triggered by a perturbation of Q by ΔQ_{pred} is computed.

The alarm count is then normalized for the total number of possible alarms and the weight of each alarm state triggered is also decreased as a function of the time delay from the initial change event to the alarm. This has the effect of focussing this measure on quickly developing cascading alarm sequences which are the most difficult to interpret and diagnose. Finally, the cascading alarms score is normalized by subtracting the background cascading alarms score. This background score is simply the cascading alarms score for no perturbation.

Computing a Total Sensor Score

We use the *surprise* score to modulate the percentile *value change* associated with a sensor. This accounts for the unusualness of a sensor value as well as the change in the sensor value that brought it to its current reading. The percentile value change score is also used to modulate the scores obtained by the causal analysis of the system: the *sensitivity* score and the *cascading alarms* score. These are modulated by the percentile *value change* because they are computed based on an analysis of the effect of a perturbation in the value of the sensor on the overall system. The remainder of the score combinations are simple sums. See Figure 2.

Application Domain

Our application domain is the hardware testbed of the water side of the Environmental Control and Life Support System (ECLSS) for Space Station Freedom. The water side of ECLSS consists of three principal systems: Multifiltration (MF), Vapor Compression and Distillation (VCD), and the Volatile Removal Assembly (VRA). Using a combination of analysis of system description documents, consultation with testbed engineers, and actual hardware testbed data, we have constructed models of all three of these subsystems. Each subsystem model contains 30-50 quantities and 15-30

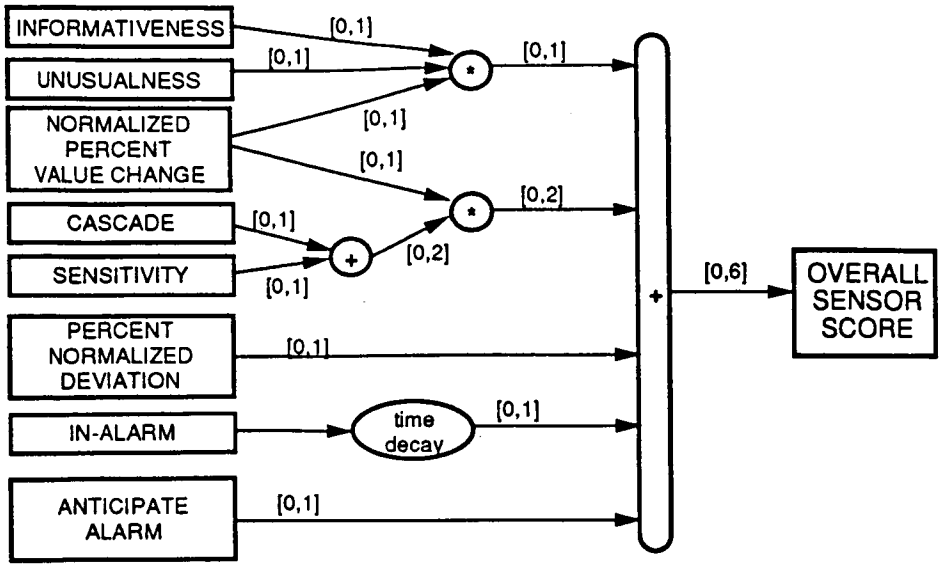


Figure 2: SELMON Sensor Scoring Algorithm.

mechanisms. Work in elaborating fault models is ongoing. This model has been validated by comparison against actual data from the subsystem testbed undergoing evaluation at the Marshall Space Flight Center (MSFC) in Huntsville, Alabama. We are also in the process of extending our model to cover the ECLSS air side subsystems.

Performance Evaluation

The output of the SELMON algorithms is dynamically computed each time the sensors are polled. SELMON produces a total ordering by importance on the set of sensors, and a window size which determines how many sensor data are presented to the operator. In order to assess whether SELMON is usefully focusing operator attention, we are comparing sensor subsets selected by SELMON to critical sensor subsets specified by domain experts as useful in understanding episodes of anomalous behavior in actual historical data from ECLSS testbed operations.

In one experiment, we asked whether or not SELMON was suppressing sensor data deemed critical by a domain expert. For this experiment, we separated the performance of the window sizing algorithm from the sensor scoring algorithm by choosing a constant window size. The specific question posed was how often did SELMON place a "critical" sensor in the top half of the sensor ordering. For a sensor set of cardinality 13, we defined the top half to be the first seven slots in the total sensor ordering. Thus the performance of a random sensor selection algorithm would be expected to be about 46.2%. Table I shows the results of this experiment. The first column identifies one of the episodes specified by the domain expert. The second column shows the number of timesteps in the episode in which the given sensor was deemed critical. The third column shows the overall SELMON "hit" rate for that episode: the number of times SELMON placed the given sensor in the top half of the sensor ordering.

EPISODE	# of timesteps	Hit Rate (%)
kc01.1	710	81.4
kf01.1	3	100
kf01.2	7	100
kf01.3	7	100
kf01.4	2	100
kf01.5	2	100
kf01.6	2	100
kf01.7	2	100
kf01.8	2	100
kf01.9	7	100
kf01.10	4	50.0
kp01.1	40	47.5
kp02.1	40	47.5
kp03.1	40	62.5
kp01.2	71	98.6
kp02.2	71	100
kp03.2	71	100
kt01.1	27	100
kt02.1	9	88.9
kt02.2	332	100
kt04.1	25	100
All	1512	87.1

Table I: SELMON performance at selecting critical sensor data.

These results suggest that SELMON performs at much better than random at replicating the attention focussing of one domain expert identifying episodes of anomalous behavior for the ECLSS testbed. SELMON's performance is not yet at the level which could

support an operational capability for real-time monitoring assistance. A more detailed analysis is ongoing to determine why SELMON performed poorly in some episodes and to examine the performance for individual sensor importance measures.

SELMON is intended to assist operators in efficient anomaly detection – the first step towards diagnosis. Another planned experiment will investigate how sensor selection supports diagnostic reasoning:

In addition to the ECLSS subsystem models which describe nominal behavior, a number of ECLSS fault models are being developed. After implementing a diagnostic reasoning algorithm, we will determine how this algorithm performs at correctly diagnosing faults from behavior traces resulting from simulation of these fault models. We will then test the performance of the diagnostic reasoning algorithm when it is given only SELMON-selected sensor data. Finally, we will test the performance of this algorithm when it is given the same number of sensor data randomly selected. Some degradation of performance is expected in the diagnostic reasoning algorithm using SELMON-selected data. A measure of success will be a significantly greater loss of performance with randomly selected data. A final caveat is that this experiment may only indirectly shed light on the ability of SELMON to support *human* troubleshooting activity.

Discussion

NASA mission operators are trained to interpret raw telemetry to create a mental model of the state of a spacecraft or spacecraft subsystem. SELMON is intended to focus operator attention on the most important sensor data. If SELMON does nothing more, it may be construed to be simply and only providing operators with less raw data to interpret, and thus may be considered to be a step in the wrong direction.

Accordingly, we recognize that an important component of the SELMON approach is the ability to provide explanations or interpretations of why a particular sensor has been placed in the monitoring window and is worthy of operator attention. Future work in the SELMON project will be oriented towards complementing focus of attention and anomaly detection capabilities with model-based interpretation capabilities.

In related work, we are also investigating the problem of sensor placement during design, using both monitorability [Chien et al 91a] and diagnosability [Chien et al 91b] criteria.

Summary

We are developing techniques to support real-time monitoring through sensor selection, the moment to moment focusing of attention on a subset of the available sensor data. Sensor selection is based on a set of importance criteria which draw on concepts from model-based reasoning and information theory. Although the SELMON project is currently targeted towards focus of human operator attention, the techniques may also support focus of attention in an autonomous monitoring and control system.

Acknowledgements

Others who have worked recently on the SELMON project include Leonard Charest and Nicolas Rouquette. We would like to thank Jay Wyatt of the Marshall Space Flight Center for many informative discussions regarding the operation of the ECLSS system.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[Chien et al 92] S. A. Chien, R. J. Doyle, and U. M. Fayyad, "Focusing Attention in Real-Time Systems Monitoring," *AAAI Spring Symposium on Selective Perception*, Stanford, March 1992.

[Chien et al 91a] S. A Chien, R. J. Doyle, and L. S. Homem de Mello, "A Model-based Reasoning Approach to Sensor Placement for Monitorability," *3rd AAAI Model-Based Reasoning Workshop*, Anaheim, July 1991.

[Chien et al 91b] S. A Chien, R. J. Doyle, and N. F. Rouquette, "A Model-based Reasoning Approach to Sensor Placement for Diagnosability," *2nd International Workshop on the Principles of Diagnosis*, Milan, October 1991.

[Doyle et al 92] R. J. Doyle, D. Berleant, L. K. Charest, Jr., U. M. Fayyad, L. S. Homem de Mello, H. J. Porta, M. D. Wiesmeyer, "Sensor Selection in Complex Systems Monitoring Using Information Quantification and Causal Reasoning," in *Recent Advanced in Qualitative Physics*, B. Faltings and P. Struss (eds.), MIT Press, *to appear.*

[Doyle and Fayyad 91] R. J. Doyle and U. M. Fayyad, "Sensor Selection Techniques in Device Monitoring," *2nd Conference on AI, Simulation and Planning in High Autonomy Systems*, Cocoa Beach, April 1991.

[Doyle et al 86] R. J. Doyle, D. J. Atkinson, and R. S. Doshi, "Generating Perception Requests and Expectations to Verify the Execution of Plans," *5th National Conference on Artificial Intelligence*, Philadelphia, August 1986.

MIXED DATA/GOAL DRIVEN INTELLIGENT REAL-TIME ASSESSMENT AND CONTROL

Bruce D'Ambrosio
Oregon State University
Computer Science Dept.
Corvallis, OR 97331-3202

We describe work in process to develop model-based systems for real-time assessment and control of complex systems. Intelligent real-time processing must balance accuracy and response-time requirements to maximize expected performance. We use belief (probability) and goal (utility) information to guide construction of an approximate system model on which we then apply approximate inference procedures to yield timely, effective assessments and control decisions.

Integration of Domain and Resource-Based Reasoning for Real-Time Control in Dynamic Environments*

Keith Morgan
Kenneth R. Whitebread
Michael Kendus
GE Advanced Technology Laboratories
Moorestown, NJ

Andrew S. Cromarty
Distributed Systems Technology
Palo Alto, CA and Tully, NY

Abstract

This paper describes a real-time software controller that successfully integrates domain-based and resource-based control reasoning to perform task execution in a dynamically changing environment. The design of the controller is based on the concept of partitioning the process to be controlled into a set of tasks, each of which achieves some process goal. It is assumed that, in general, there are multiple ways (tasks) to achieve a goal. The controller dynamically determines current goals and their current criticality, choosing and scheduling tasks to achieve those goals in the time available. It incorporates rule-based goal reasoning, a TMS-based criticality propagation mechanism, and a real-time scheduler. The controller has been used to build a knowledge-based situation assessment system that formed a major component of a real-time, distributed, cooperative problem solving system built under DARPA contract. It is also being employed in other applications now in progress.

1 Background

The results reported in this paper were derived in the course of developing the Situation Assessment (SA) component of the DARPA Submarine Operational Automation System (SOAS). The SOAS project explored the application of advanced automation techniques such as AI to the support of the commander of an attack submarine. It produced a large-scale distributed software system whose components provided support in such functions as tactical planning and situation assessment.

To control the course-depth-speed profile of the vessel (referred to as "ownship"), manage its staff, and allocate its weapons and other resources, the Commanding Officer (CO) must maintain a timely and accurate understanding of the external situation. SA's responsibility is to construct and maintain, in coordination with a human operator, a *scene assessment* in real time. The scene is generated by processing and interpreting sensor data (primarily sonar data) to create a representation of the submarine's physical environment and the various man-made objects (ships, other submarines, etc., which are known as *contacts*) within range of the sub's sensors.

Knowledge-based processing is used in SA due to the voluminous amount of highly uncertain and incomplete information it must intelligently analyze to achieve its goals. Among the key problems that had to be solved in the design and implementation of SA was the development of an effective method of *domain-based real-time control of problem solving*. This paper describes the SA control architecture which was developed to meet SA's control requirements. By treating problem solving control as a reasoning problem based on both domain knowledge and knowledge of computing resources, the SA control (called the "Metacontroller" succeeded in managing the complex task of situation assessment in real time.

2 Statement of the Problem

The technical challenges to effective problem-solving control for SA stem from four major factors:

- The system must operate in a real-time, dynamic environment.
- The system must select the most useful assessment tasks under current circumstances.
- The system must maintain consistency of its goals and actions (both current and planned) as new data alters perception of the situation.
- The system must support cooperative problem-solving (with its human operator and with other components).

This section explains each of these requirements and their implications for the design of the Metacontroller.

The submarine situation assessment problem is inherently real-time. Ownship must respond to external events such as possible collisions with other vessels. The CO must also consider time-constraints imposed by external circumstances in planning and executing actions which he initiates such as carrying out an attack. Since assessment is a prerequisite to sensible and effective vessel management and target prosecution, the time constraints of the larger command problem devolve on to assessment as well.

Because of the volume of incoming sensor data, the number of assessment tasks which could sensibly be pursued at any time is typically too large to allow execution

*Sponsored by Defense Advanced Research Projects Agency, DARPA ASTO/STP, Submarine Operational Automation System ARPA Order No.6661/50, Issued by DARPA/CMO under Contract MDA972-90-C-0005

of all such tasks. This information overload is one of the reasons for developing an assessment aid for the submarine command staff in the first place. The Metacontroller must therefore guide its own assessment activities on a moment-to-moment timescale based on the expected impact of pursuing one *vs.* another assessment task. For example, an explicit decision must be made concerning whether the next few milliseconds of computing time should be spent on pursuing possible contact correlations or whether a popup contact is a collision threat. Such decisions depend both on time constraints (if there there is a collision threat, is it imminent?) and on domain knowledge (is further analysis of a given contact likely to be of importance to ownship's mission or safety?).

The data which drive situation assessment are often noisy, unreliable, incomplete, unavailable, evolving, or even conflicting, due to sensor limitations, operator limitations, inherent sensor error, physical vessel limitations, and data processing limitations. The Metacontroller must therefore be capable of adapting its evaluation of the priority of assessment tasks as new sensor input is received.

SA is conceived as an operator-controlled component of a distributed problem solving system. It therefore cannot behave autonomously even though it is expected to intelligently control its own functions. Thus, SA control must accept and integrate with its self-derived problem-solving goals both directives from the operator and requests from other intelligent components.

The major challenge in designing the Metacontroller was the *simultaneous* satisfaction of these requirements by the design. Each of these topics has been researched, and treated in isolation. For example, methods for reasoning about task time constraints have been studied, but our application required that such reasoning be integrated with domain-based reasoning (essentially expert reasoning) since the choice of which assessment tasks to pursue depends both on time constraints and on knowledge about which elements of the current situation are operationally most important.

3 Approach

The approach treats control of the situation assessment reasoning process as itself a reasoning process carried on at a metalevel with respect to reasoning done on the domain. The assessment process is conceived as a set of domain-level reasoning tasks (e.g., the task of determining whether a given contact poses a collision threat). The control of that process is conceived of as a reasoning task whose purpose is to decide which domain- level task to perform next. The control reasoning process reasons about goals and tasks. Its function is to determine current system goals, their criticality, and which tasks should be performed given current goals. The reasoning process takes into account the effects of new input data, time constraints, and estimates of the time required for task completion.

We found that this control model could be implemented using conventional AI techniques for the various functions required. The following list indicates the methods used and their purpose:

1. Use domain expert as source of domain-based control knowledge. (much of what the expert said was about control)
2. Divide all functions into tasks supporting goals
3. Use goal-based tasking to perform all system and domain functions
4. Use rule-based reasoning to determine current system goals and tasks
5. Ensure consistency of goals (hence, of the current and planned course of action) through the use of a truth maintenance function.
6. Establish deadlines for all real-time tasks and use these deadlines to schedule the tasks for execution.
7. Trade-off competing real-time tasks according to deadline and precedence.
8. Trade-off competing non-real-time tasks according to criticality.

The remainder of this section presents a description of the system.

3.1 Overview of the Control Architecture

The SA component is composed of the Metacontroller and a set of procedures for performing the domain-level reasoning tasks which create an assessment from sensor input data.

The top-level architecture of the Metacontroller is depicted in Figure 1. The principal elements of this architecture are *Event generation*, the *Metaplanner*, the *Scheduler*, the *Executor*, and the *Truth Maintenance System* with their associated data and knowledge bases and queues.

Event Generation signals the controller, when an event has occurred. An event may occurs when new information is sent to SA via its communication system, or when SA's own domain-reasoning infers a tactically significant datum that could affect SA's control decisions..

The *Metaplanner* uses expert control knowledge to establish and modify goals in response to the events signalled by event generation. It also creates tasks from events. Additionally, the metaplanner is responsible for establishing the class of a task (real- or non-real-time) and initialization of relevant parameters of the task, e.g. its deadline.

The *Scheduler* produces a total ordering of all tasks in accordance with the constraints implied by the class and parameters of the tasks. The scheduler uses a real-time control policy to schedule a real-time task before a non-real- time task, regardless of the relative criticality of those tasks.

The *Executor* executes the task on the top of the schedule. It is also responsible for collecting run-time statistics on each of the task to assist in accurate scheduling of future tasks.

The Metacontroller's *Truth Maintenance System* maintains the consistency of the Metaplanner's goal and task reasoning as new input data is received.

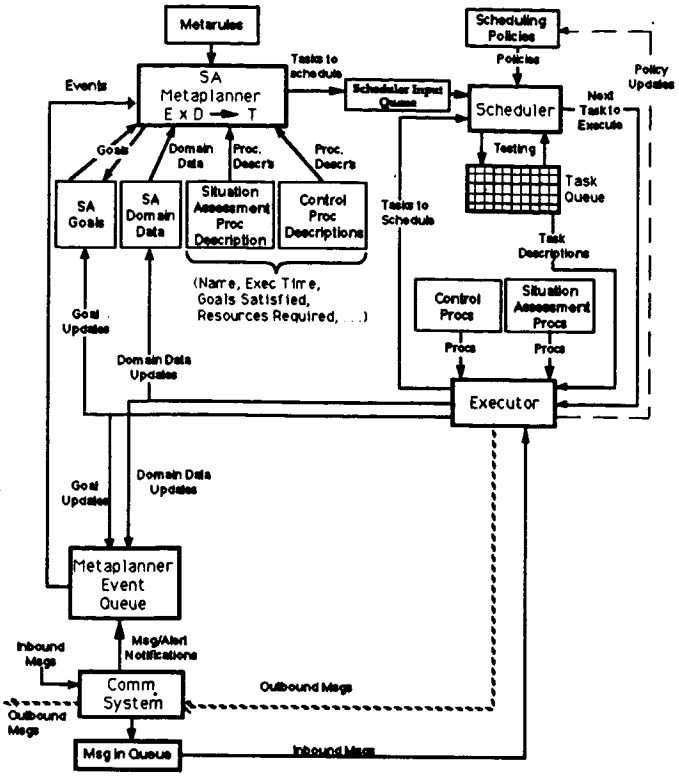


Figure 1: SA Architecture

The tasks whose execution the Metacontroller governs fall into two categories: *control procedure* and *domain procedures*. Domain procedures perform operations fulfilling SA's assessment function. For example, a task that attempts to correlate a newly received sonar tonal with a known contact is classed as a domain procedure. Control procedures perform the operations required to implement the control process itself. For example, a procedure that reads new messages from an input port is a control procedure. Both domain and control procedures are identified for execution by the Metaplanner and scheduled by the Scheduler for execution. Thus, the Metacontroller controls both the execution of SA's problem-solving process and its own behavior as an algorithm.

We now discuss in detail the role of each of the Metacontroller components.

3.2 Distributed Event Generation

Inbound messages to SA are received by the Communications System Interface. The receipt of a message results in the generation of an SA *event*, with the type (class) of event signalled depending on the class of received message.

When the Communications System Interface signals a message receipt event, it also places the body of the received message in the Message InQueue, where it is held until an appropriate control procedure dequeues and processes the message.

Outbound messages are created by individual domain procedures or control procedures and are passed through the Communications System Interface for forwarding to other SOAS components.

3.3 Metaplanner

The Metaplanner can be thought of as a transition function that maps events and goals into tasks (or goals). Periodically, events in the event queue are dequeued *en masse* and processed by the Metaplanner. The Metaplanner employs a dynamic goal base, the events from the event queue, and a set of metalevel planning rules *(metarules)* to select actions (if any) appropriate to each event dequeued from the Metaplanner Event Queue. The Metaplanner produces sets of tasks for execution by the Scheduler, based on the current set of events, goals, and metarules.

Upon each invocation of the Metaplanner, events are read from the Event Queue, goals from the goal base, and metarules relating goals and events to tasks from the Metarule knowledge base; the Metaplanner then forward chains exhaustively on the events to determine the full set of consequent tasks that the events (taken as assertions) "imply."

Metarules take the form

$$\mathcal{E}\mathcal{D} \rightarrow \mathcal{T}$$

where

$\mathcal{E}$ is a set of Events dequeued from the Metaplanner Event Queue

$\mathcal{D}$ is the control Data set, consisting of dynamic, persistent goals and of control *facts*. Control facts are assertions of beliefs currently held about the domain, for example, "Contact S2 could be a submarine".

$\mathcal{T}$ is a set of generated tasks

that is, they map event-goal pairs to a corresponding *task* to execute. The task may be either a base-level (domain level) or metalevel (control level) task, and it is through control level tasks that the goal base is modified and new goals are established. For example, a metarule might be of the form, "If event E_i has occurred and we have goal G_j, then schedule for execution task T_g," where T_g is a task that inserts a new goal g into the goal base.[1]

The Metaplanner forward chains on the events fully until all eligible metarules have fired. Each generated task is assigned a *criticality* derived from the goal that motivated its elaboration. The set of tasks to execute resulting from a single such Metaplanner invocation are queued and held in the Scheduler Input Queue; when the closure of the event set has been computed for this goal and rule set, the Metaplanner flushes the Event Queue, queues its set of derived tasks in the Scheduler Input Queue, and expires.

Although events do not live beyond one Metaplanner invocation, metarules and goals are maintained across Meta-

[1] Note that by requiring goal knowledge base manipulations to occur singly within the context of a task execution rather than at Metaplanning time, we effectively eliminate the order-dependent processing ambiguity that could arise in Metaplanner operations if it were possible to modify the goal set while it was in active use by the Metaplanner.

planner invocations. The metarule and goal knowledge bases may be modified from time to time by control procedures as described above. Most such instances of control procedure execution as well as most executions of domain procedures occur because the Metaplanner identifies the procedures as tasks to be scheduled and executed as already described. However, as Figure 1 indicates, tasks may also be submitted for scheduling directly from the Executor. This occurs when an executing task spawns a child task. Task spawning was implemented as an efficient way to handle such features as the establishment of periodic tasks and partitioning tasks into subtasks. When a task spawns a child task, the parent may assign any goal to the child. The assigned goal is added to the goal base if necessary. In practise, most children are assigned the goal of the parent. Note that task creation of goals and tasks does not violate the conceptual model already described. The same effect could have been achieved by instead implementing tasks which signal an appropriate event instead of directly spawning a new task. The event could then be processed with metarules added to the knowledge base for the purpose. The decision to use direct spawning of tasks simply allowed more efficient implementation.

Metaplanning, essentially the process of deciding what to plan, is an approximate description of the activities engaged in by this, the largest part of the Metacontroller. Whereas the SA Scheduler actually determines which computing activities SA will engage in during the next processing epoch, the SA Metaplanner determines what the Scheduler will have available as its scheduling alternatives. Viewed alternatively, the Metaplanner engages in *explicit planning at the metalevel*, that is, it employs a knowledge-based planning technique to determine what sets of tasks SA should execute as a computational system.

3.4 Scheduler

The Scheduler is a true real-time task scheduler. Each task is a descriptor pairing a procedure identifier with invocation specific parameters. Associated with each procedure in the Procedure Description knowledge bases consulted by the Metaplanner are data characterizing the procedure's anticipated execution time; the Scheduler uses these execution time predictions together with the Metaplanner-specified criticality value assigned to each task to build an execution schedule for all pending tasks, including both the set of tasks dequeued from the Scheduler Input Queue at Scheduler invocation time and the set of previously queued but unexecuted pending tasks already waiting in the Scheduler's Task Queue.

The Scheduling policy is as follows. Each task has a *criticality*, nominally inherited from the parent goal that led to the task's elaboration by the Metaplanner.[2] Most tasks also have a real-time deadline; these are *real-time (RT) tasks*, *vs.* the *non-real-time (non-RT) tasks*, which lack execution deadlines. (It is required that an estimate of execution time be available for tasks having execution deadlines.)

Real-time tasks generally are scheduled to execute before non-real-time tasks, to ensure that real-time execution criteria are satisfied.[3] The RT tasks are scheduled for execution (i.e. mapped onto an execution timeline) based on their required completion time and anticipated execution time. Where two tasks qualify for the same timeslot, criticality is used as a tie-breaker. Once such a schedule has been constructed for RT tasks, non-RT tasks are scheduled; where possible, they are fitted into the interstices between already-scheduled RT tasks, and if no such space exists in the schedule, they are appended to the end of the schedule in order of decreasing criticality.

At the completion of a Scheduler invocation, the Task Queue is a total ordering of tasks ordered by scheduled start time. The first task in the Task Queue is the *next task to execute*.

3.5 Executor

The Executor dequeues the *next task to execute* and assigns the processor to it. Execution is non-preemptive, i.e. the task executes to completion once initiated. From SA's perspective, all task executions in the current prototype are atomic in the sense that they always run to completion and cannot be interrupted by (for example) intervening message deliveries or other internal or external events. The question of whether to implement task interruption was considered early in the process of designing the Metacontroller. Analysis of both options showed that task interruption would vastly increase the complexity of the goal and task reasoning process. We therefore chose to define atomic tasks.

3.6 Data and Knowledge Bases

In addition to the major components of the architecture as described above, SA also contains several data and knowledge bases. They include:

- The *Metarules knowledge base*, from which the Metaplanner obtains its rules for planning task executions based on the current contact (as evidenced by triggered events) and the current goal set.

[2] In some cases, as described above, a task will be presented to the Scheduler for scheduling and subsequent execution by another task. The creating task is permitted to specify any of several criticalities for the created task: that of the creating task, that of some goal to which the created task is intended to bear a causal relationship, or the same criticality as some other identified task in which the new class is intended to have equivalence-class membership for criticality purposes. Under most circumstances, however, we would expect that tasks will be elaborated by the metaplanner and will inherit their criticality from the parent goal that they are being executed to satisfy.

[3] This follows from the principal that SA is a real-time problem-solving system. If satisfaction of execution deadlines were subordinated to some other metric, e.g. task "priority," then SA would be a conventional non-real-time problem-solving system. In a real-time system, the deadlines are incontrovertibly more important than other performance metrics.

- The *SA Goals knowledge base*, containing the assessment goals that will employed by the metalevel to perform metaplanning.
- The *SA Domain Database*, comprising the collection of facts, assertions, and hypotheses about the undersea world that are used to produce assessments. This includes all knowledge concerning the current set of hypothetical contacts in the external world.
- The *Situation Assessment Procedure Description database*, containing a description of each computational procedure that SA's Executor can execute to manipulate domain-level data (i.e. data about vessels, contacts, acoustics, etc.).
- The *Control Procedure Description database*, wherein reside corresponding descriptions of each metalevel computational procedure (those that manipulate input messages, task queues, etc.).
- The *Situation Assessment Procedures database*, containing the bodies of executable domain-level procedures, such as procedures for performing correlation of contacts.
- The *Control Procedures database*, containing the actual bodies of executable metalevel procedures.

3.7 Truth Maintenance Substrate

SA uses a truth maintenance system (TMS) to ensure consistency between goals. The SA TMS is an extension of the JTMS (Justification-based TMS) [6; 2] approach, in which justifications are represented as Horne clauses. The JTMS has been extended to establish and propagate goal criticalities. This capability is essential for reasoning about the importance of goals, and in turn, of tasks.

The TMS represents two types of "assertions": facts and goals. Facts are statements of belief that ultimately justify one or more goals, goals are a state SA is trying to achieve or a question to answer. Similarly justifications are of two types: fact and goal (meaning that the TMS justifies the in-ness of a fact or goal).

A fact justification conjoins a set of facts and goals (i.e. assertions) and represents boolean support for the consequent fact when the conjunction is true. A goal justification conjoins a set of facts and goals to represent boolean support but also represents the importance (criticality) that support wants to lend to the consequent goal. Each goal justification has an associated criticality.

The criticality of a goal is assigned in one of two ways. If the goal is an assumed goal (i.e., a premise) a criticality is assigned at the time of the assumption. Other goals are derived goals, and obtain their criticality from the criticality of justifications. The criticality of the supporting justification is used as the criticality of the goal. When a goal has more than one justification, the one with the maximum criticality is used to assign the criticality.

4 Test Results

TheMetacontroller was tested as part of the SOAS prototype. Tests were performed in a distributed computing environment consisting of networked workstations. Each major component of SOAS, e.g., the SA component, resided on a single workstation and communicated with other components on other workstations through a communication substrate based on ISIS. An additional workstation on the network supported a simulator which exercised the SOAS prototype by providing simulations of: sensor data, data on ownship (e.g., ownship course and speed), and environmental data such as sound-velocity profiles. This distributed system was tested against several contact-behavior scenarios by personnel with both computer science and Navy operational experience.

Through the use of the Metacontroller, SA was able to meet about 80% of its real-time deadlines in the midst of executing optional tasks opportunistically. We expect that the 80% figure could be improved upon through improved use of the existing metacontroller, for example, by replacing unbounded-time algorithms with any-time versions.

SA uses about 30 metarules and typically maintains goal sets containing between 20 and 100 goals. The metacontroller runs efficiently. Approximately 5% of the total processing time typical SA scenarios is spent in metacontroller processing. The metacontroller is implemented in Common Lisp; we plan to port the metacontroller to the C language.

5 Conclusions

The development of the Metacontroller demonstrates two points:

- A feasible method for integrating domain and resource-based reasoning about control of a knowledge-based procedure.
- The practicality of creating solutions to difficult AI system design problems by coupling several well-known AI techniques.

An obvious problem in building real-time AI systems is the fact that control of the system typically depends on two largely unrelated factors:

- Time deadlines and related computing resource constraints.
- The control decisions inherent in the problem-solving domain.

Each of these factors has been the subject of investigation, individually. The first factor has been studied in some depth [5]. Concepts such as any-time algorithms and the use of interrupts in AI tasks have been investigated [3; 7]. The second factor has led to well-known concepts such as blackboard systems and goal-based reasoning. Although the general problem of integrating these two factors has been discussed (e.g., [4]), there appear to be few published sources describing detailed mechanisms for such combined reasoning. Typically, much of the expertise in a given problem domain has to do with reasoning about what to do next.

This inference process cannot be discarded when real-time constraints are imposed. Instead, it must be integrated with the complimentary process of reasoning about deadlines. The Metacontroller design provides a mechanism to do this integration.

The second point demonstrated, that several AI techniques can be usefully coupled, is important in advancing the success of AI as an engineering discipline. The Metacontroller combines:

- Rule-based reasoning to form goals,
- Goal-directed processing,
- TMS-based criticality propagation,
- Real-time scheduler.

The noncontrol elements of SA integrate additional techniques as well.

While each of these techniques is based on well-known concepts, its not typical to find them coupled in a single integrated function. More frequently, we see systems built using one or two techniques (e.g., a diagnostic system built around a TMS [1]). Such homogeneous designs are appropriate for basic research in AI techniques and for engineering applications where they happen to work. However, our experience has led us to believe that the engineering potential of the existing body of AI research cannot be properly exploited unless multiple techniques are integrated. In the case of the Metacontroller, we showed that integration of multiple techniques produced very promising results on a problem which appeared beyond the means of any single techniques with which the authors are familiar.

References

[1] de Kleer, Johan, and Williams, Brian, Diagnosing multiple faults, *Artificial Intelligence*, 32:97–130, 1987.

[2] Doyle, Jon, A Truth Maintenance System, *Artificial Intelligence*, 12:231–272, 1979.

[3] Erman, Lee D. ,Ed., *Intelligent Real-Time Problem Solving: Workshop Report*, Santa Cruz, CA, November 8 and 9, 1989.

[4] Hayes-Roth, B., Washington, R., Hewett, R., Hewett, M., Seiver, A, Intelligent Real-Time Monitoring and Control, Technical Report No. KSL 89-05, Knowledge Systems Laboratory, Stanford University, Stanford, CA.

[5] Laffey, Thomas J., Cox, Preston A., Schmidt, James L., Kao, Simon N., Read, Jackson Y., Real-Time Knowledge-Based Systems, *AI Magazine*, Spring, 1988, Vol. 9, No. 1, pp. 27-45.

[6] McAllester, David, A Three-Valued Truth Maintenance System S.B. Thesis, Department of Electrical Engineering, MIT, Cambridge, MA, 1978.

[7] Sharma, D.D., and Narayan, Srini, An Architecture for Intelligent Task Interruption, *Proceedings of the Workshop on Real-Time Artificial Intelligence Problems*, Detroit, MI, August 20, 1989.

TECHNICAL DEVELOPMENTS AND AUTOMATED LAUNCH PROCESSING ADVISORY SYSTEM

Barbara Brown
NASA/Kennedy Space Center
Kennedy Space Center, FL 32899

Abstract unavailable at time of publication.

EOS DIAGNOSTICS OF AN IN DEVELOPMENT SYSTEM FOR EARTH OBSERVING SYSTEM MONITORING AND DIAGNOSTICS AT GODDARD

Bob Dominy
NASA/Goddard Space Flight Center
Greenbelt Road
Greenbelt, MD 20771

Abstract unavailable at time of publication.

Session A5: DIAGNOSTICS AND ANALYSIS

Session Chair: James Villarreal

AN ON-LINE EXPERT SYSTEM FOR DIAGNOSING ENVIRONMENTALLY INDUCED SPACECRAFT ANOMALIES USING CLIPS

by

Mark Rolincik, University Research Foundation, Greenbelt, MD 20771
Michael Lauriente, NASA Goddard Space Flight Center, Code 420, Greenbelt, MD 20771
Harry C. Koons & David Gorney, The Aerospace Corporation, P.O.Box 92957, LosAngeles, CA 90009

ABSTRACT

A new rule-based, expert system for diagnosing spacecraft anomalies is under development. The knowledge base consists of over two-hundred (200) rules and provides links to historical and environmental databases. Environmental causes considered are bulk charging, single event upsets (SEU), surface charging, and total radiation dose.

The system's driver translates forward chaining rules into a backward chaining sequence, prompting the user for information pertinent to the causes considered. The use of heuristics frees the user from searching through large amounts of irrelevant information and allows the user to input partial information (varying degrees of confidence in an answer) or 'unknown' to any question.

The expert system not only provides scientists with needed risk analysis and confidence estimates not available in standard numerical models or databases but is also an effective learning tool. In addition, the architecture of the expert system allows easy additions to the knowledge base and the database. For example, new frames concerning orbital debris and ionospheric scintillation are being considered. The system currently runs on a MicroVAX and uses C Language Integrated Production System (CLIPS), an expert shell developed by the NASA Johnson Center AI Laboratory in Houston.

BACKGROUND

The Air Force (1) and NASA (2) jointly are designing a new rule-based, on-line expert system for diagnosing in-flight spacecraft anomalies. This system provides an effective method for saving knowledge and allows computers to sift through large amounts of data, homing in on significant information. Most importantly, it uses heuristics in addition to algorithms which allows approximate reasoning and inference, and the ability to attack problems not rigidly defined.

The modularity of the expert system allows for easy updates and modifications. It not only provides scientists with needed risk analysis and confidence not found in the usual programs, but it is also an effective learning tool, and the window implementation makes it very easy to use. The system currently runs on a microVAX II at Goddard space Flight Center (GSFC). The inference engine used is NASA's C Language Integrated Production System (CLIPS) (3). CLIPS is not only compatible with both C and Fortran languages, but it has features which include the ability to compile the rules and save them in a binary image file, thus allowing faster execution than a typical rule interpretive system. This feature qualifies CLIPS to be used as an expert shell, i.e., an environment where the rules can reside and be accessed. The expert system is divided into frames, something most programmers would call "modules,"

and each frame relates to one of the causes of the satellite anomaly.

DESCRIPTION

The knowledge base consists of over two-hundred (200) rules and provides links to historical and environmental databases. Initially, recognized experts in the field were queried on how to diagnose anomalies. The "rules of thumb" they provided were formatted into logical rules. The system output was verified by referring to historical case studies and historical data. The architecture of the system was designed to emulate the way the user normally looks at data to diagnose anomalies. The expert system not only consolidates expertise in a uniform, objective, and logical way, but it also offers "smart" ways of accessing various databases which are transparent to the user. Then by applying various rules in its knowledge base, the system is queried, as appropriate, to arrive at a conclusion.

The current version of the Space Environment Anomalies Expert system (SEAES) is able to attribute the causes of satellite anomalies to one of several possible categories, including surface charging, bulk charging, single event upsets (SEU), and total radiation dose. ("Unknown" is also a possible and plausible conclusion, depending on the quantity of data available. The architecture of SEAS is such that other causes could be added if a satisfactory rule base were developed. Some examples that have been considered are ionospheric scintillation (e.g., pertinent to commanding errors or telemetry link failures) and orbital debris (pertinent to mechanical breakups or damage). Rule bases and data bases are being compiled for each of these categories, and these new frames will be added to the SEAES after verification and testing has been completed. The system goes through a "decision tree" based on these rules in order to arrive at the likely cause of anomaly. The rule base includes the expert system rules that will be "fired" under control of the inference engine and entered in a defined "if-then" format. The user interface links to databases which include past environmental data, satellite data and previous known anomalies. Information regarding satellite design, specifications and orbital history need to be assimilated with previous anomalies data and environmental conditions, while addressing the specific circumstances of individual users.

NEW FRAMES

As an example of our approach to the addition of new frames, consider the case of orbital debris diagnosis. The rationale for including orbital debris in an analysis of satellite anomalies is that debris is an ever-increasing threat to spacecraft. The effects of orbital debris on spacecraft range from minor erosion of surfaces to more severe mechanical damage or even breakup in the case of collisions with large objects. From a system design standpoint, it is useful to understand the cause of a mechanical breakup. For example, breakups can be caused by internal component ruptures or explosions of pressurized systems such as fuel, attitude control gases, or batteries. Design changes would be called for in these cases, while design mitigation would not be appropriate for collisional breakups. While orbital debris data bases offer some guidelines for assessing the probabilities of collisions for spacecraft, they do not offer any insight into a particular occurrence of a breakup. An expert system would be able to help the user interpret the available data bases in terms of the particular anomaly under study. Furthermore, it is possible to examine orbital debris on the resulting fragments to specifically identify the cause of the breakup as being due to collision or explosion.

A common and useful data display for understanding satellite breakups is the Gabbard diagram(4), Figure 1. The Gabbard diagram plots an apogee and perigee height against its orbital period for each of the trackable fragments following breakup. In an elliptical orbit a Gabbard diagram will have two points: The apogee and perigee heights aligned above its orbital period. To denote apogee, "x" symbols are used and "+" symbols are used for perigee. In a circular orbit, the Gabbard diagram is a single point for each fragment. Figure 1, shows a Gabbard diagram, plotting the apogee and perigee heights versus orbital period for fragments following breakup. The distribution, symmetry and scatter of the points can all be used in analysis of the event. These rules can be incorporated into a knowledge

Sample Gabbard Diagram

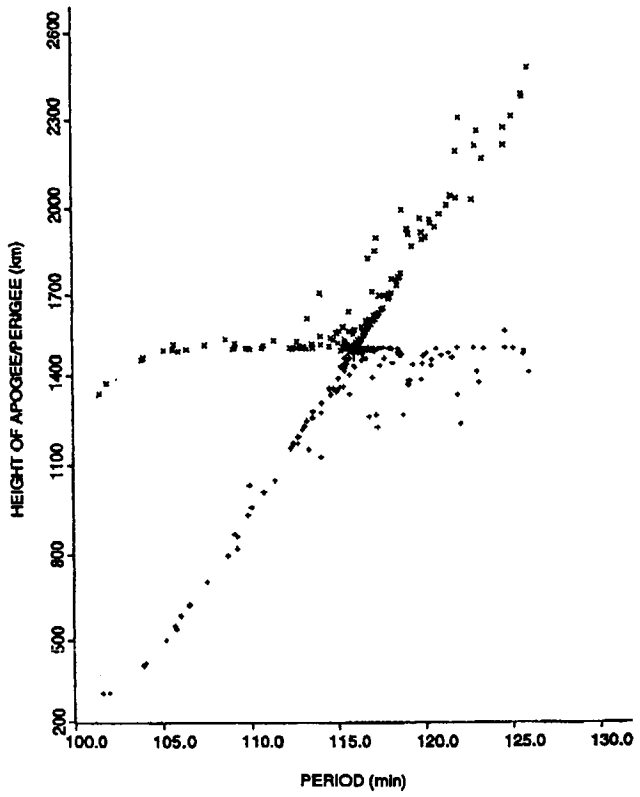


Figure 1. Gabbard Diagram [Johnson and McKnight (4)]

base which, when applied to actual data can be used to assess a breakup.

A sample exponential fit to Gabbard diagram:

If b<2.0 then CAUSE=COLLISION CF25
If b>2.0 then CAUSE=EXPLOSION CF25

Perform polynomial fit to Gabbard diagram:
If A_1 <.15 then CAUSE=COLLISION CF25
If A_1 >.15 then CAUSE=EXPLOSION CF25

Dispersion of large pieces: If any fragments larger than 1 m^2 are dispersed over 50% of the total range of fragments, then CAUSE=EXPLOSION CF10 ELSE CAUSE=COLLISION CF10.

Asymmetry of large fragments: If fragments larger than 1 m^2 are asymmetrically distributed about the parent, then CAUSE=EXPLOSION CF15 ELSE CAUSE=COLLISION CF15.

Orderedness of dispersion: If the Gabbard diagram is very ordered, then CAUSE=COLLISION CF15 ELSE CAUSE=EXPLOSION CF15.

Velocity analysis: If average velocity imparted decreases as fragment size increases, then CAUSE=COLLISION CF10 ELSE CAUSE=EXPLOSION CF10.

A set of rules as these can be added easily as a separate frame of the expert system. Similarly, rule sets for other causes, such as ionospheric scintillation or others, can be added as well.

RESEARCH TOOL

The on-line feature was considered a natural communication tool for educating the users on this innovative venture. In addition, the opportunity was there for the users to feedback information to improve on the system. The key to advancement in this endeavor is communication between users. The user here is either a forecaster, a scientist, an engineer, an operator, or perhaps a contractor, who needs to know something about the effects of the environment on a satellite or a satellite subsystem, recognizing that they will have access to a variety of databases and knowledge. As of the present, we call it a "research system." That is a technical name for an expert system at a specific state of development beyond the prototype stage, where it has been shown to produce useful answers. It doesn't contain all the possible rules it could, but it is getting close to being ready for other people to start evaluating it. We are interested in granting accounts to users for the purpose of evaluation.

KNOWLEDGE BASE

Unlike its algorithmic predecessors, an expert system can be flexible in the way that it attacks complex problems. By virtue of its three basic parts (a knowledge base, a fact base, and a driver interface) an expert system

more closely simulates the methods of human experts who use a combination of known, empirically derived formulae, hunches based on degrees of certainty and experience, and even judicious "fudging" when specific data is lacking. Figure 2 shows the expert system configuration.

The knowledge base, with its set of rules, is what makes a rule-based expert system unique. Best thought of as an independent collection of "if...then" statements, the rules are created by experts in their respective fields and reflect the current level of human experience, along with its uncertainties. Under the weight of these rules, and by the use of multi-field variables, an expert system

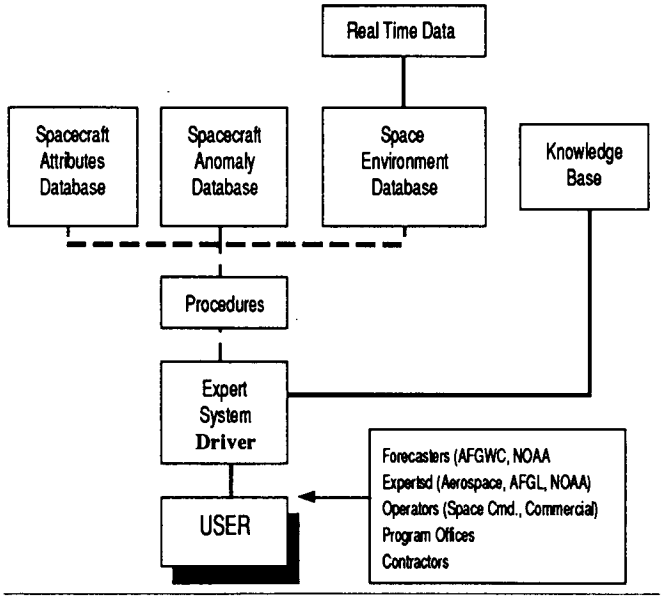


Figure 2. Expert System Configuration

can be said to "ponder the possibilities" presented by the databases and current knowledge which are too extensive to be readily assimilated by any single person. Rather than being limited to conclusions that must satisfy a set of tightly ordered mathematical statements, the system is free to offer suggestions, considerations, and likelihoods.

The rule format used in the expert system is shown in Figure 3. Each rule has a subject associated with it (in this case one of the four causes considered), a description of the rule, and then the actual rule itself. The rules also have what is termed a 'confidence factor' associated with the right hand side of each rule. Algorithms, which normal programs are limited to using, have a

```
    RULE201
    =======
      SUBJECT :: BULK_CHARGING-RULES
      DESCRIPTION :: (recurs when fluence high)
If    1) the recurrence of the anomaly, and
      2) the recurrence is OF_HIGH_PENETRATING_FLUX, and
      3)  1) the seven-day accumulated fluence of penetrating electrons is
             HIGH, or
          2) the seven-day accumulated fluence of penetrating electrons is
             VERY_HIGH,
Then there is suggestive evidence (60%) that the cause of the anomaly is
BULK_CHARGING.

      IF   :: (RECURRENCE AND PERIODICITY = OF_HIGH_PENETRATING_FLUX AND
              (ACCUM_FLUEN = HIGH OR ACCUM_FLUEN = VERY_HIGH ) )
      THEN :: (CAUSE = BULK_CHARGING CF 60)



    RULE110
    =======
      SUBJECT :: TOTAL_DOSE-RULES
      DESCRIPTION :: (Local time recurrence rules out total radiation dose.
If    1) the recurrence of the anomaly, and
      2) the recurrence of an anomaly in a specific local-time sector.,
Then it is definite (100%) that the cause of the anomaly is not TOTAL_DOSE.

      IF   :: (RECURRENCE AND LT_RECUR)
      THEN :: (CAUSE != TOTAL_DOSE)
```

Figure 3. Rule Format

100% certainty to them, and are a subset of the general heuristic rules which the expert system uses.

This aspect of the rule-based expert system is very important in diagnosing anomalous behavior since much of the knowledge, rules and experience required to diagnose these anomalies have confidence factors associated with them. The use of such confidence factors in the expert system introduces the concept of 'risk assessment' to the diagnostic procedure and the inclusion of knowledge which otherwise would be lost, since it is, at the very least, extremely difficult to represent such knowledge using mathematical formulae.

VARIABLES

SEAES' use of variables is another area which makes this system unique, allowing it to handle non-algorithmic, equivocal problems. A variable in this system can take on one of three settings. It can be 'unset', meaning that it has not been input by the user and that no rule has been able to determine a value for it; it can be 'unknown' which means the user was prompted for the variable but did not know it; or it can have one or more 'values'. The unique aspects of the

system are that not only can the expert system continue to execute when variables are unknown, but when variables do have values, each value has a confidence factor associated with it. Figure 4 shows examples of variable formats.

```
INCLINATION
==========
 TRANSLATION :: (the inclination of the plane of the orbit with respect
                 to the earth's equatorial plane )
 PROMPT :: (Select the inclination of the satellite with respect to the
            earth's equatorial plane. )
 TYPE :: SINGLEVALUED
 EXPECT :: (EQUATORIAL LOW_INCLINATION HIGH_INCLINATION POLAR OTHER)
 UPDATED-BY :: (RULE041 RULE133 RULE134 RULE135 RULE136 RULE132 RULE138
                RULE139 RULE140 RULE141 RULE142 RULE137 )
 ANTECEDENT-BY :: (RULE026 RULE030)
 USED-BY :: (RULE017 RULE016 RULE091 RULE089)
 HELP :: ("Low inclination orbits are below 30 deg.  High
           inclination orbits are above 60 deg.  Polar orbits
           are above 80 deg.  Interplanetary orbits are undefined." )
 CERTAINTY-FACTOR-RANGE :: UNKNOWN


LT_RECUR
========
 TRANSLATION :: (the recurrence of an anomaly in a specific local-time
                 sector. )
 PROMPT :: ("Indicate the degree of certainty that you have that this
             type of anomaly has a strong tendency to recur in one local
             time sector, for example the nightside or the dayside of the
             earth?" )
 TYPE :: YES/NO
 USED-BY :: (RULE019 RULE020 RULE110 RULE054 RULE188 RULE189 RULE190
             RULE191 RULE192 RULE193 RULE194 RULE043 )
 HELP :: (The anomaly should have occurred a few times (i.e. six or more)
          before you have confidence that the recurrence is related to a
          specific local-time sector.  Generally we are asking if the
          anomaly has a very strong tendency to occur within a 12 hour range
          in local time. )
 CERTAINTY-FACTOR-RANGE :: POSITIVE
```

Figure 4. Variable Format

In the variable format, the translation and prompt string are self-explanatory. Each variable also has a type associated with it, either 'single-valued', 'multi-valued', or 'yes/no'. The 'expect' field is a list of the possible values for that variable which the user can select when and if he/she is prompted for that variable. The 'updated_by' field is a list of rules which are able to determine values for that variable, while the 'used_by' field contains rules which require this variable in order to fire. (It is possible that in order for a rule to fire, a variable must be 'unknown'). The 'help' field is the information displayed when the user presses the help key, requesting more information on the variable being prompted for. The 'certainty-factor-range' (CFR) is particular to this system, and can have a value of 'unknown', 'positive', or 'full'. The CFR being 'unknown' means that this is a possible input for that variable. If the CFR is 'positive', the user can input degrees of confidence from 0 to 100 for each of the inputted values for that variable. Finally, if the CFR is 'full' the user can input degrees of confidence from -100 to 100 which mean a range from being 100% certain the variable is not a specific value to being 100% certain that the variable is a specific value.

The confidence factors relay the confidence the user has in a certain value of the variable. This is very important since there is most likely information of which the user is not 100% sure. Such information is lost in normal programs. The combination of the confidence factors of variables and those of the rules propagates the confidence factors to other variables which are determined by these rules and ultimately to the cause of the anomaly.

Figure 5 shows an input screen for a single-valued variable (which assumes 100% confidence), and a CFR of 'unknown'. Figure 6 is an example of the input screen for a multi-valued variable with a 'positive' CFR. Notice how the variable in figure 6 can have more than one value, and each value has its own confidence factor associated with it.

FACT BASE

The fact base, a collection of informative sources related to the topic of interest, is the second basic part of an expert system. It can consist of as many separate data bases as may be deemed pertinent to solving the problem at hand. In the case of spacecraft anomalies, a fact base might contain information on the hardware currently in use, other active and past satellite systems, and historical data for orbital environments.

The database selection screen is shown in Figure 7, which shows the databases available for this system along with an example of the expert system help facility which is available for any variable. An important advantage obtained in using the expert system is that once it has been established which databases are available, the rules determine which information is pertinent, access the database for the relevant information and apply this information, (all of which is transparent to the user). Also, the database accessing is modular and easily expandable, thus if more databases need to be added,

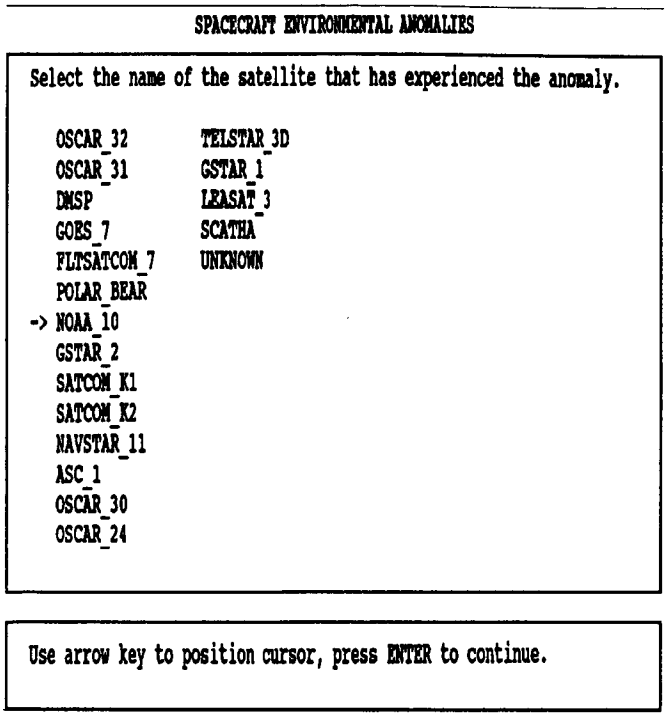


Figure 5. Satellite Selection

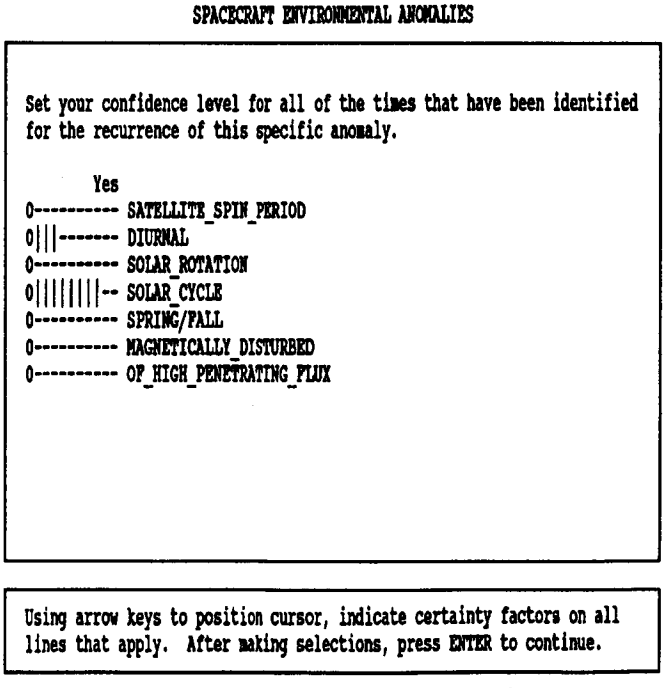


Figure 6. Multi-valued input with confidence

only the selection screen needs to be changed, and the new rules added to the knowledge base. These capabilities free the user from sifting through large amounts of data and ensure that only pertinent information and all pertinent information is used in the diagnosis.

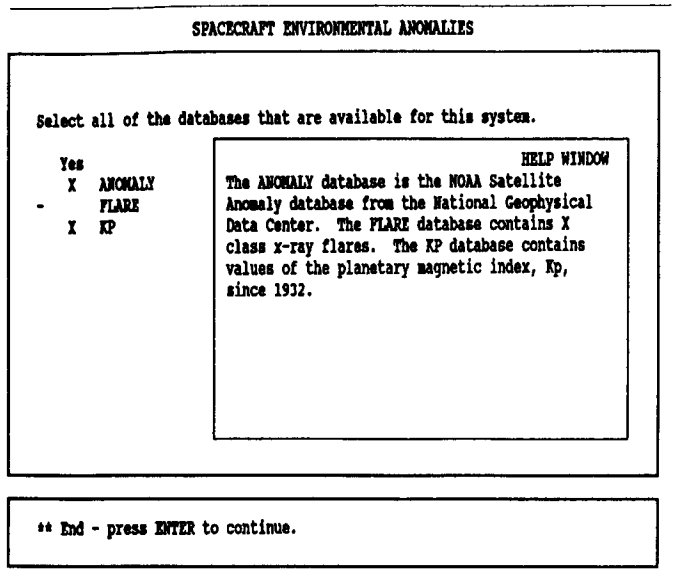


Figure 7. Database selection screen

INTERFACE

The interface is one of the aspects which makes all expert systems different from one another. Since the expert shell, databases and knowledge base are independent and modular, the main purpose of the interface is to create a coordinating system which is not only user friendly, but also provides the necessary features to assist the user in understanding the system and the results.

The system's current interface driver translates forward chaining rules into a backward chaining sequence, prompting the user for information pertinent to the causes he/she wishes to consider. The main purpose of the driver is to maintain information regarding the variables which are being determined, the rules which can determine these variables, the status of the variables, and which rules can be fired.

Some variables are designated as initial variables or goal variables. The system first prompts the user for the initial variables. The driver then stacks the goal variables on the run time stack and searches the knowledge base for rules which determine (or 'update') these variables, and then puts them on the stack as well. The system focuses on those possibilities of high confidence and then assists the user by directing him/her to areas of consideration that directly affect the particular problem.

The goal (variable) in our system is the CAUSE of the anomaly, a multi-valued field variable with a 'full' CFR, since it can take on any number of the four possible causes where each cause has its own confidence factor associated with it ranging from -100 to 100.

If a variable on the left hand side of a stacked rule is unset, this variable becomes the current goal variable and is put on the stack, and the process continues. If a variable is on the stack and has not determined by any rules, or by the available database, and it has a prompt string, the user is prompted for it. This can be thought of as a transformation of the forward chaining rules in the knowledge base into a backward chaining variable sequence. Once a variable has a value, it is removed from the stack and the rules which use this variable are fired, discarded, or require the driver to put the next variable on that rule's left hand side onto the stack. The chaining process continues until the stack is empty.

Any rule on the stack that can be fired does so transparently to the user, where the confidence factors of the individual variables on its left hand side (LHS) are used for determining the confidence or validity of the entire LHS. When a rule fires, it executes the right hand side (RHS), and the confidence factor associated with its LHS is used in conjunction with the confidence factor of the rule to propagate the confidence to the RHS. This RHS execution can entail the setting of variables, the use of mathematical calculations, or the accessing of databases.

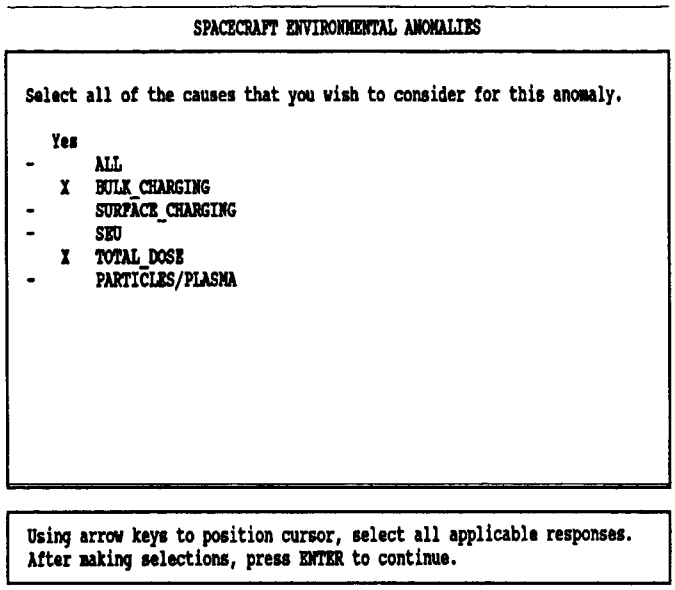


Figure 8. Causes selection screen

LEARNING TOOL

One of the most beneficial aspects of the system is its use as a learning tool for diagnosing spacecraft anomalies. A user is initially given a choice between either 'novice' or 'expert' mode for the current session. If the user selects the novice mode the system automatically gives detailed explanations and descriptions of terms and reasoning as the session progresses, in a sense teaching the user about the topic or topics. The expert mode, on the other hand, simply executes the session without giving these extra explanations, unless the user specifically requests them.

The user is also given the option of selecting which causes are to be considered. (See figure 8) This selection determines a knowledge base sub-group, so that only rules in this specific environmental area are considered. In this way the user can learn what variables, information and data affect, and are important to, that cause. In addition to this, in the features described next, the user is actually able to access the relevant rules him/herself and other variables and facts which were determined by using these rules.

The ability to add intricate features and options is primarily due to the modularity of the system which the expert shell and expert system knowledge base concept itself provide. These features are the most impressive in demonstrating the capabilities of the EnviroNET expert system and its advantages over the usual, strictly mathematical, programming techniques.

The user interface also provides for accessing graphics. For example, if the user inputs that one of the databases available is Kp, the system will ask if he/she wishes to see the Kp historical graph for the time around which the anomaly occurred. If the input is 'yes', then a graph similar to the one shown in figure 9 will be displayed. (If ,however, the date is 'unset', then the system will first ask for it, and if the date is 'unknown' the system will ignore this line of questioning altogether.) This gives the user a much needed overall view of environmental information and conditions around the date in question.

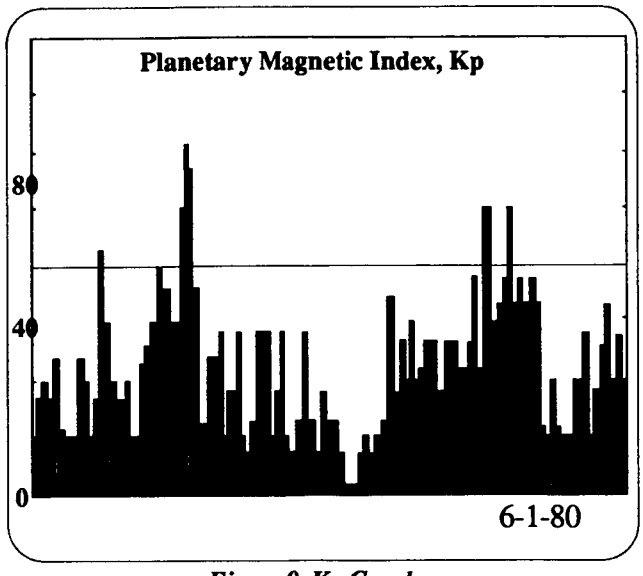


Figure 9. Kp Graph

UNIQUE FEATURES

Another feature which makes the expert system unique is its trace capability. The user can turn on the trace and send it to the screen or a file. The trace shows the rules as they are tested, variables as they are pushed onto the run time stack and determined, and searches of the databases. (See figure 10) This allows the user to understand what is happening at any step and see the

```
Setting  TOTAL_DOSE_TECHNOLOGY  =  AMORPHOUS_TTL  cf  100
Testing  RULE119
RULE119  FAILS
Testing  RULE128
RULE128  FAILS
Testing  RULE129
RULE129  FAILS
Testing  RULE131
Applying  RULE131
Setting  TOTAL_DOSE_THRESHOLD  =  1000000  cf  100
Testing  RULE120
Applying  RULE120
Setting  CAUSE  =  TOTAL_DOSE  cf  -86
old cf  -30
Mark_antec_rules_for  CAUSE  RULE027
Try_marked_antec_rules
Testing  RULE027


** More - press ENTER to continue.
```

Figure 10. Trace example

knowledge that is being used, thus giving the user confidence in the system. This type of capability is obviously not available in the purely algorithmic programs. Due to the amount of information the user could be prompted for and depending on the particular session, the user may want to review his/her inputs. This capability is available in the 'REVIEW' facility. This option also provides the user with a simple way of comparing different inputs of different sessions.

A feature which demonstrates a definite advantage of the rule-based expert system is what is called the 'WHY' option. Any time the system prompts the user for a variable, the user can ask the expert system why the system needs this variable. The system then uses its run time stack (a backward chaining stack) to follow and show the reasoning backward to the goal; that is, the cause of the anomaly. Figures 11-12 show an example of this. This is not only vital to understanding and

```
                    SPACECRAFT ENVIRONMENTAL ANOMALIES

Enter a value between 0 and 400 for the maximum value of the planetary

                                                          WHY WINDOW

the three hour planetary index Ap is needed to determine the level of
magnetic activity in the magnetosphere

RULE094
If   the three hour planetary index Ap is greater than 30,
Then it is definite (100%) that the level of magnetic activity in the
magnetosphere is DISTURBED.

** More - press ENTER to continue.
```

Figure 11. Backward reasoning

having confidence in the system, but it also is an important part of the expert system's use as a learning tool.

A final feature which sets the expert system apart is the 'HOW' command. As with all programs, the expert system is constantly determining variables by means other than the user inputting them, whether by the heuristics and algorithms in the rules or by extracting values from the databases. This command allows the users to, at any time, see what variables have been

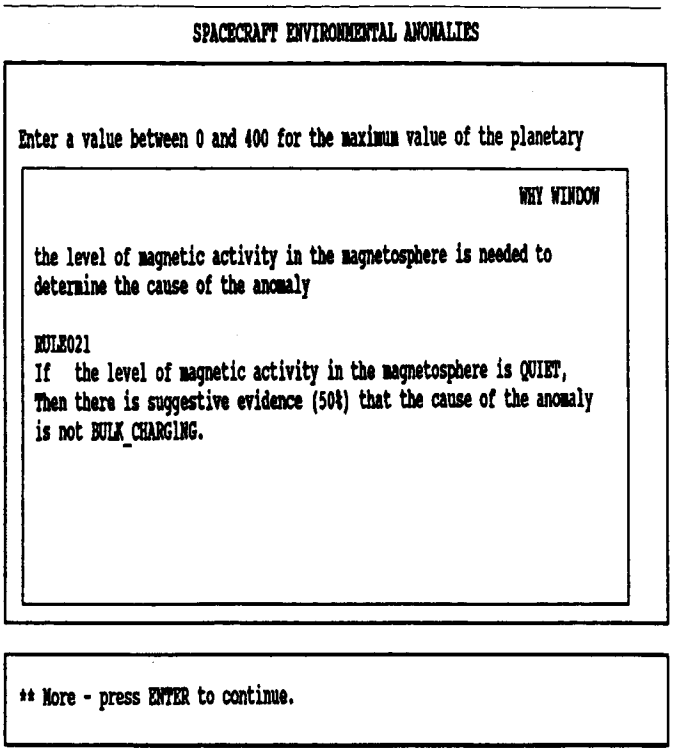


Figure 12. Backward reasoning (con't.)

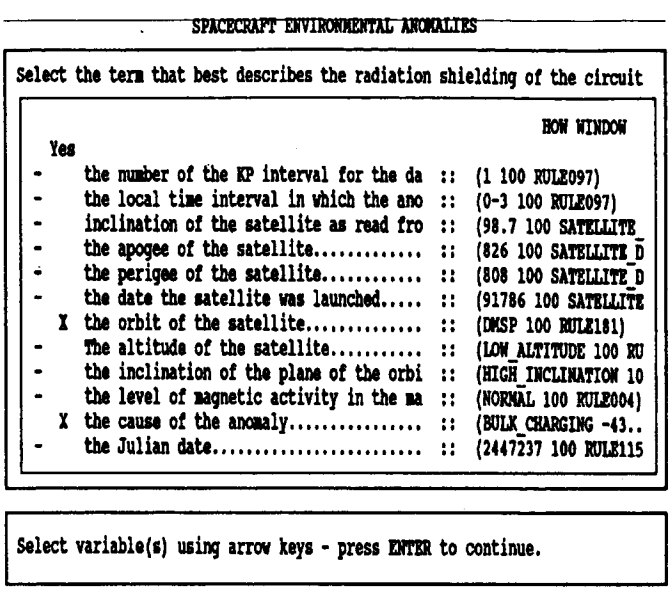


Figure 13. HOW facility

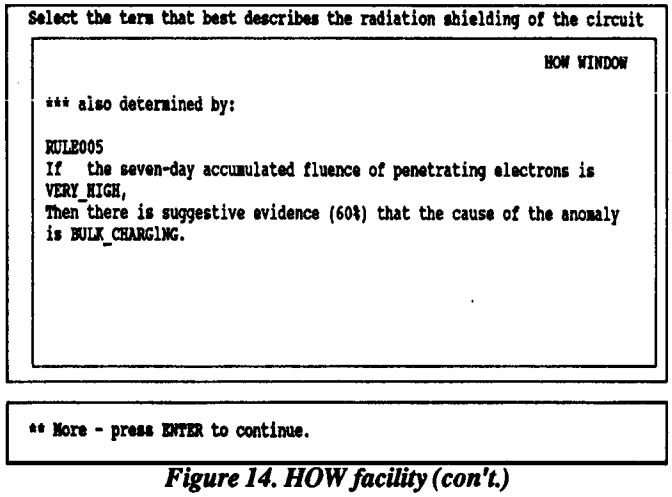


Figure 14. HOW facility (con't.)

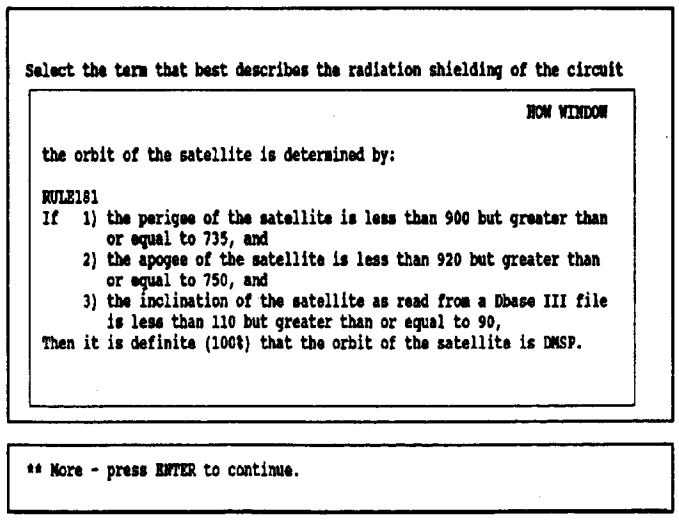


Figure 15. HOW facility (con't.)

determined by means other than user input, their values, and which rules (or databases) were used to determine them. Figures 13-15 show an example of this feature. The user first selects which variables he/she wants to look at and then the system proceeds to show which rules determined them. Notice how it is possible for variables to be determined (or updated) by more than one rule. The user of course can choose any number of variables, though for this example only one variable, the cause of the anomaly, was selected. This feature not only gives the user complete control over the system, but allows him/her to see all the facts and knowledge that can be inferred from the inputs they have given, the available databases, and the expertise in the rules. As a final option, the user is also allowed, at any point, to exit from the program or begin a new session without ever leaving the program's window screen.

RESULTS

The diagnostic results are in the form of confidence factors derived from both the confidence assigned to rules by the experts and also the confidence of variables

input by the user. Both the confidence in the rules/ heuristics and the input of certainty factors by the user are needed to diagnose anomalies as they contain vital knowledge which can only be represented as such. The results window is shown in Figure 16.

The results window in our system includes, in addition to the cause(s) of the anomaly, the orbit of the satellite, whether input by the user or determined by rules, and a list of the causes considered in the diagnostics. The window can easily be modified to display any other information which is considered important. In the example, the cause of the anomaly was determined to be bulk charging with a confidence of 64%, and determined not to be total radiation dose with a confidence of 80%. The knowledge base does, of course, contain rules and formulae which can determine the cause of the anomaly with 100% confidence, or completely rule out a particular cause. For these situations the system will simply say that the cause, for example, is bulk charging or is not total dose.

The main concern with the system is the actual confidence and validity of the rules themselves. Since experts in any field are likely to disagree over certain areas, there may be rules to which other experts would apply slightly higher or lower degrees of confidence. This is certainly a consideration when using such a system, though it must be remembered that it is due to such a confidence/certainty question in the field that this type of expert system is needed. In general, as more quantitative environmental data becomes available in the immediate area of a spacecraft, we can apply the higher confidences to all of the system's rules. In addition, the features provided by the interface allow the user to see exactly what rules are being used so there is complete awareness and understanding of the formulae and knowledge being used.

An advantage of this particular system is that its interface is completely generic. Not only can the system run on many machines, the interface can be used in any field since the rules and knowledge base are completely independent of it. By substituting rules from another field, the system becomes an expert system for that field able to diagnose or solve problems towards which its tailored rules converge. In this sense the software is completely reusable.

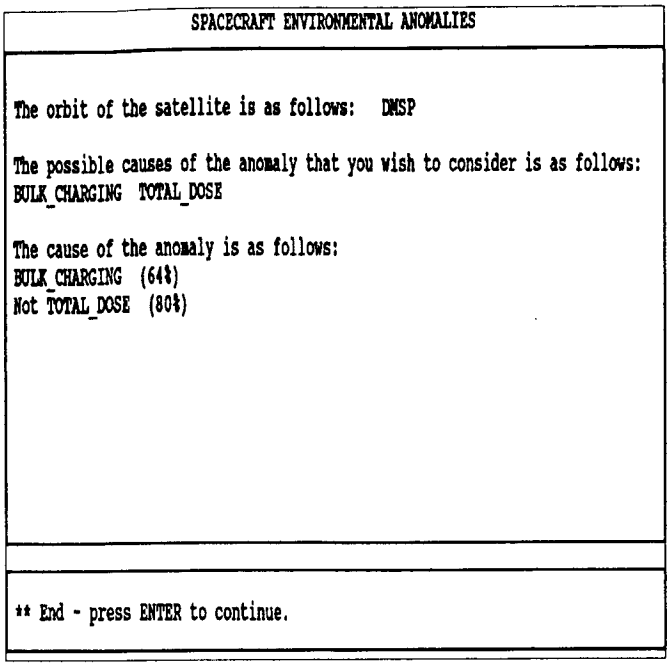


Figure 16. Results screen

FUTURE WORK

We are improving our EnviroNET network with the addition of an IBM RISC 6000. Once there, not only will the speed of the Expert System be increased, but with the use of X Windows the system will also be enhanced.

For example, with X Windows the user could have one query window which prompts him/her for information, another separate window that displays which rules are being tested and fired, which variables are being searched for, and another window for graphics. With these multiple windows the user can see the entire system working at once and be freed from having to change windows to see system information.

CONCLUSION

SEAES combines the algorithmic capabilities of mathematical programs and diagnostic models with expert heuristic knowledge, and uses confidence factors in variables and rules to calculate results with degrees of human confidence associated with them. Since the causes of environmentally induced spacecraft anomalies depend not only on algorithms, but also on

environmental conditions, rules and information the conclusion can rarely be known with 100% certainty. Based on present experiences, the role for the expert system is for either quasi-real time, or post analysis. There is a need to greatly improve our ability to predict the environment before meaningful work can be done in forecasting satellite anomalies.

ACKNOWLEDGMENTS

We are indebted to the staff of NASA's Johnson Space Center's AI Laboratory for their cooperation in the use of CLIPS. Funding was provided by NASA's Office of Safety and Mission Quality, the U.S. Air Force's Phillips Laboratory and Space Systems Division under contract F04701-88-C-0089

REFERENCES

1.Koons, H. C., and Gorney, D. J., "Spacecraft Environmental Anomalies Expert System: A Status Report," Aerospace Report # ATR-88 (9562)-1, The Aerospace Corporation, El Segundo, CA., 1 Dec. 1988.

2.EnviroNET: An On-line Environmental Interactive Resource, Proc. Fourth Annual Space Operations and Research (SOAR) Symposium, June 26-28, 1990

3.Giarratano, J. C. (1989, May). "Artificial Intelligence Section," CLIPS User's Guide, Version 4.3 of CLIPS, Lyndon B. Johnson Space Center.

4.Johnson, Nicholas L. and Darren S. McKnight, "Artificial Space Debris", Orbit Book Company, Malabar FL, 1987.

USING MACHINE LEARNING TECHNIQUES TO AUTOMATE SKY SURVEY CATALOG GENERATION

Usama M. Fayyad†, Nicholas Weir‡, J.C. Roden†, S.G. Djorgovski‡, and R.J. Doyle†

†Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

‡Department of Astronomy
California Institute of Technology
Pasadena, CA 91125

ABSTRACT

We describe the application of machine classification techniques to the development of an automated tool for the reduction of a large scientific data set. The 2nd Palomar Observatory Sky Survey provides comprehensive photographic coverage of the northern celestial hemisphere. The photographic plates are being digitized into images containing on the order of 10^7 galaxies and 10^8 stars. Since the size of this data set precludes manual analysis and classification of objects, our approach is to develop a software system which integrates independently developed techniques for image processing and data classification. Image processing routines are applied to identify and measure features of sky objects. Selected features are used to determine the classification of each object. GID3* and O-BTree, two inductive learning techniques, are used to automatically learn classification decision trees from examples. We describe the techniques used, the details of our specific application, and the initial encouraging results which indicate that our approach is well-suited to the problem. The benefits of the approach are increased data reduction throughput, consistency of classification, and the automated derivation of classifications rules that will form an objective, examinable basis for classifying sky objects. Furthermore, astronomers will be freed from the tedium of an intensely visual task to pursue more challenging analysis and interpretation problems given automatically catalogued data.

1. INTRODUCTION

In this paper we present an application of machine learning techniques to the automation of the task of cataloguing sky objects in digitized sky images. The Sky Image Cataloging and Analysis Tool (SkICAT) is being developed for use on the images resulting from the 2nd Palomar Observatory Sky Survey (POSS-II) conducted by the California Institute of Technology (Caltech). The photographic plates collected from the survey are being digitized at the Space Telescope Science Institute (STScI). This process will result in about 1788 digital images of roughly $23{,}000^2$ pixels each. Each image is expected to contain on the order 10^5 sky objects.

The first step in analyzing the results of a sky survey is to identify, measure, and catalog the detected objects in the image into their respective classes. Once the objects have been classified, further scientific analysis can proceed. For example, the resulting catalog may be used to test models of the formation of large-scale structure in the universe, probe galactic structure from star counts, perform automatic identifications of radio or infrared sources, and so forth. The task of reducing the images to catalog entries is a laborious time-consuming process. A manual approach to constructing the catalog implies that many scientists need to expend large amounts of time on a visually intensive task that may involve significant subjective judgment. The goal of our project is to automate the process, thus alleviating the burden of cataloguing objects from the scientist and providing a more objective methodology for reducing the data sets. Another goal of this work is to classify objects whose intensity (isophotal magnitude) is too faint for recognition by inspection, hence requiring an automated classification procedure. Faint objects constitute the majority of objects on any given plate. We plan to automate the classification of objects that are at least one magnitude fainter than objects classified in previous surveys using comparable photographic material.

The goals of this paper are:

1. to introduce the machine learning techniques used and emphasize their general applicability to other data reduction or diagnostic classification tasks, and
2. to give a general, high-level description of the current application domain.

We therefore do not provide the details of either the learning algorithms or the technical aspects of the domain. We aim to point out an instance where the learning algorithms proved to be useful and powerful tool in the automation of scientific data analysis.

2. MACHINE LEARNING BACKGROUND

One of the goals of Artificial Intelligence (AI) research is to provide mechanisms for emulating human decision-making and problem solving capabilities, using computer programs. The first AI attempts at such systems appeared as part of the technology known as "expert systems". However, serious difficulties arose that pointed out the difficulties in endowing a system with sufficient knowledge to execute a specific task. The first such difficulty is the "knowledge acquisition bottleneck" [Feig81] due to experts finding it difficult to express their knowledge, or explain their actions, in terms of concise situation-action rules. The second problem arises in a different situation: What if a task is not well-understood, even by the experts in that area? Many processes are not well-understood and thus even experts cannot predict the outputs for a given set of inputs. An example of this situation is manifested in previous experience with the automation of the reactive ion etching (RIE) process in semiconductor manufacturing [Chen90]. In such domains, abundant data are available from the experiments conducted, or items produced. However, models that relate how output variables are affected by changes in the controlling variables are not available. Experts strongly rely on familiarity with the data and on "intuitive" knowledge of such a domain. How would one go about constructing an expert system for such an application?

The machine learning approach to circumventing the aforementioned hurdles calls for extracting classification rules from data directly. Rather than require that a domain expert provide knowledge, the learning algorithm attempts to discover, or induce, rules that emulate expert decisions in different circumstances by observing examples of expert tasks. The basic approach is described in the next section.

Two other reasons exist for the need for a machine learning approach. The first is the growing number of large diagnostic and scientific databases. A database that stores instances of diagnostic tasks is typically accessed by keyword or condition lookup. As the size of the database grows, such an approach becomes ineffective since a query may easily return hundreds of matches making simple case-based usage impractical. For large scientific databases the problem is to search for and detect patterns of interest, or to perform pre-processing necessary for subsequent analysis. Sizes are now becoming too large for manual processing. Learning techniques can serve as effective tools for aiding in the analysis, reduction, and visualization of large scientific databases. Another motivation is the evolution of complex systems that have an error detection capability. Communication networks are an example. Faults are detectable by the network hardware. Several thousand faults may occur during a day. To debug such a network, a human would need to sift through large amounts of data in search of an explanation. An automated capability of deriving conditions under which certain faults occur may be of great help to the engineer in uncovering underlying problems in the hardware.

2.1. THE MACHINE LEARNING APPROACH

The machine learning approach prescribes inducing classifiers by automatically analyzing classified examples rather than interviewing domain experts. A training example consists of a description of a situation and the action performed by the expert in that situation. The situation is described in terms of a set of attributes or variables. An attribute may be *continuous* (numerical) or *discrete* (nominal). For example, a nominal attribute may be *shape* with values {*square, triangle, circle* }. Examples of continuous attribute are *pressure, area,* or *temperature.* The action associated with the situation, the *class* to which the example belongs, is a specification of one of a fixed set of pre-defined classes. The class of each training example is typically determined by a human expert during normal task execution. Example actions (classes) in a diagnostic setting may be *raise temperature, decrease pressure, accept batch,...* If the classes represent diagnostic actions, then the classification problem becomes equivalent to diagnosis. The goal of the learning program is to derive conditions, expressed in terms of the attributes, that are predictive of the classes. Such rules may then be used by an expert system to classify future examples. Of course, the quality of the rules depends on the validity of the conditions chosen to predict each action.

A training example is a vector of attribute values along with the class to which the example belongs. Assume there are m attributes $A_1,...,A_m$, p classes $C_1,...,C_p$. A training example is an $m+1$-tuple $\langle b_1,b_2,...,b_m ; C_j\rangle$, where each b_i is one of the values of the attribute A_i: $\{a_{i1},...,a_{ir_i}\}$, and C_j is one of the p classes. A rule for predicting some class C_j consists of a specification of the values of one or more attributes on the left hand side and a specification of the class on the right hand side.

For example, consider the simplified small example set shown in Table 1. It consists of six examples e-1 through e-6. There are two attributes: S and L. The attributes can take the values *low, normal,* and *high.*

example	S	L	class
e-1	normal	normal	*PH*
e-2	normal	high	*PL*
e-3	high	high	*PL*
e-4	high	low	*PH*
e-5	low	normal	*FRL*
e-6	low	high	*FRL*

Table 1: A Simple Training Set.

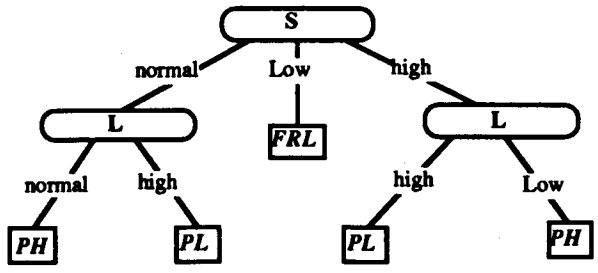


Figure 1: Decision Tree Generated by ID3 for Data Set of Table 1.

There are three classes: *FRH*, *PL*, and *PH*. A simple rule consistent with these examples may be:

IF (S = low) THEN class is *FRL*

Note that this is only an illustrative simplification. Typically, the number of examples of a meaningful training set is at least in the hundreds, while the number of attributes is usually in the tens.

2.2. INDUCING RULES FROM TRAINING EXAMPLES

Assume that there are m attributes as described above. Let each attribute A_i take on one of r_i values $\{ a_{i1},..., a_{ir_i} \}$. Assuming that on average an attribute takes on one of r values, there are $p(r+1)^m$ possible rules for predicting the p classes. It is computationally infeasible for a program to explore the space of all possible classification rules. In general, the problem of determining the minimal set of rules that cover a training set is NP-hard. Hence, there is no known computationally feasible (polynomial) algorithm for finding the solution. It is therefore likely that a heuristic solution to the problem is the only feasible one. A particularly efficient method for extracting rules from data is to generate a decision tree [Brei84, Quin86]. A decision tree consists of nodes that are tests on the attributes. The outgoing branches of a node correspond to all the possible outcomes of the test at the node. The examples at a node in the tree are thus *partitioned* along the branches and each child node gets its corresponding subset of examples. A popular algorithm for generating decision trees is Quinlan's ID3 [Quin86], now commercially available.

ID3 starts by placing all the training examples at the root node of the tree. An attribute is selected to partition the data. For each value of the attribute, a branch is created and the corresponding subset of examples that have the attribute value specified by the branch are moved to the newly created child node. The algorithm is applied recursively to each child node until either all examples at a node are of one class, or all the examples at that node have the same values for all the attributes. An example decision tree generated by ID3 for the sample data set given in Table 1 is shown in Figure 1.

Every leaf in the decision tree represents a classification rule. The path from the root of the tree to a leaf determines the conditions of the corresponding rule. The class at the leaf represents the rule's action.

Note that the critical decision in such a top-down decision tree generation algorithm is the choice of attribute at a node. The attribute selection is based on minimizing an information entropy measure applied to the examples at a node. The measure favors attributes that result in partitioning the data into subsets that have low class entropy. A subset of data has low class entropy when the majority of examples in it belong to a single class. The algorithm basically chooses the attribute that provides the locally maximum degree of discrimination between classes. For a detailed discussion of the information entropy minimization selection criterion see [Quin86, Fayy91].

3. OVERCOMING PROBLEMS WITH ID3 TREES

It is beyond the scope of this paper to discuss the details of the ID3 algorithm and the criterion used to select the next attribute to branch on. The criterion for choosing the attribute clearly determines whether a "good" or "bad" tree is generated by the algorithm[1]. Since making the optimal attribute choice is computationally infeasible, ID3 utilizes a heuristic criterion which favors the attribute that results in the

[1] See [Fayy90,Fayy91] for the details of what we formally mean by one decision tree being better than another.

partition having the least information entropy with respect to the classes. This is generally a good criterion and often results in relatively good choices. However, there are weaknesses inherent in the ID3 algorithm that are due mainly to the fact that it creates a branch for each value of the attribute chosen for branching.

3.1. PROBLEMS WITH ID3 TREES

Let an attribute A, with values { $a_1,a_2,...,a_r$ } be selected for branching. ID3 will partition the data along r branches each representing one of the values of A. However, it might be the case that only values a_1 and a_2 are of relevance to the classification task while the rest of the values may not have any special predictive value for the classes. These extra branches are harmful in three ways:

1. They result in rules that are overspecialized. The leaf nodes that are the descendants of the nodes created by the extraneous branches will be conditioned on particular irrelevant attribute values.
2. They unnecessarily partition the data, thus reducing the number of examples at each child node. The subsequent attribute choices made at such child nodes will be based on an unjustifiably reduced subset of data. The quality of such choices is thus unnecessarily reduced.
3. They increase the likelihood of occurrence of the missing branches problem. This problem occurs because not every possible combination of attribute values is present in the examples.

The third problem can be illustrated in the ID3 tree shown in Figure 1. Consider two possible unclassified examples which are to be classified by the tree of Figure 1:

ex1: (S = low) & (L = low)
ex2: (S = normal) & (L = low)

Both ex1 and ex2 are examples that have combinations of attribute values that did not appear in the training set of Table 1. However, the tree readily classifies ex1 as being of class *FRL*, but ex2 fails to be classified by the tree. This is because the subtree under the (S = normal) branch has no branch for (L = low). We shall shortly illustrate how the occurrence of *missing branches* may be avoided.

The main problem with the tree of Figure 1 is that the normal and high S branches should not be separated. Low S is the only value of relevance to the occurrence of a *LFR* event. It would be desirable if the learning algorithm could somehow take account of such situations by avoiding branching on attribute values that are not individually relevant. This would reduce the occurrence of the three problems listed above.

3.2. ALGORITHMS GID3* AND O-BTREE

As discussed earlier, improving the tree generation algorithm will improve the classification accuracy of the produced classifier. To avoid some of the problems described in the previous section, we developed the GID3* algorithm [Fayy91]. We generalized the ID3 algorithm so that it does not necessarily branch on each value of the chosen attribute. GID3* can branch on arbitrary individual values of an attribute and "lump" the rest of the values in a single *default branch*. Unlike the other branches of the tree which represent a single value, the default branch represents a subset of values of an attribute. Unnecessary subdivision of the data may thus be reduced. Figure 2 shows the tree GID3* would generate for the data set of Table 1. Note that both examples, ex1 and ex2, are classifiable by this tree. The missing branch problem that prevented the tree of Figure 1 from classifying ex2 does not occur in the tree of Figure 2.

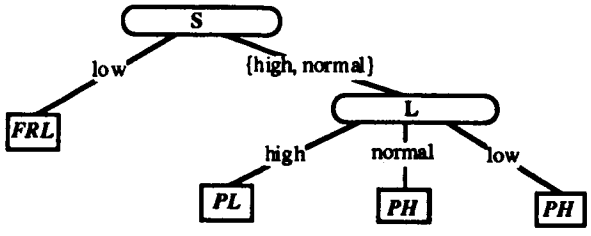


Figure 2: Decision Tree generated by GID3* for Data of Table 1.

The other learning algorithm we use is O-BTree [Fayy91,Fayy92b]. Unlike ID3 and GID3* whose attribute selection criterion is based on information entropy, O-BTree's selection criterion employs a measure from a different family of selection measures that measure class separation rather than class impurity (as in entropy). For a detailed discussion of GID3* and O-BTree as well as extensive empirical demonstration of their comparative performance, see [Fayy91].

Note that in our discussion and examples we assumed that attributes are discrete. Numerical attributes are discretized prior to attribute selection at each node. The range of a numerical attribute is partitioned into

several intervals thus creating a (temporary) discrete attribute. For a detailed discussion of attribute discretization see [Fayy91, Fayy92a]. Interested readers are referred to [Fayy91] for detailed accounts of the algorithms, the attribute selection criteria, the weaknesses of the ID3 approach, and various performance measures to evaluate the quality of the resulting trees. Performance measures include error rate in classifying new examples and measures of the size complexity of the generated tree.

We turn our attention to the task of automating sky object classification. We then present our results that demonstrate that O-BTree and GID3* performed significantly better than ID3 for this domain.

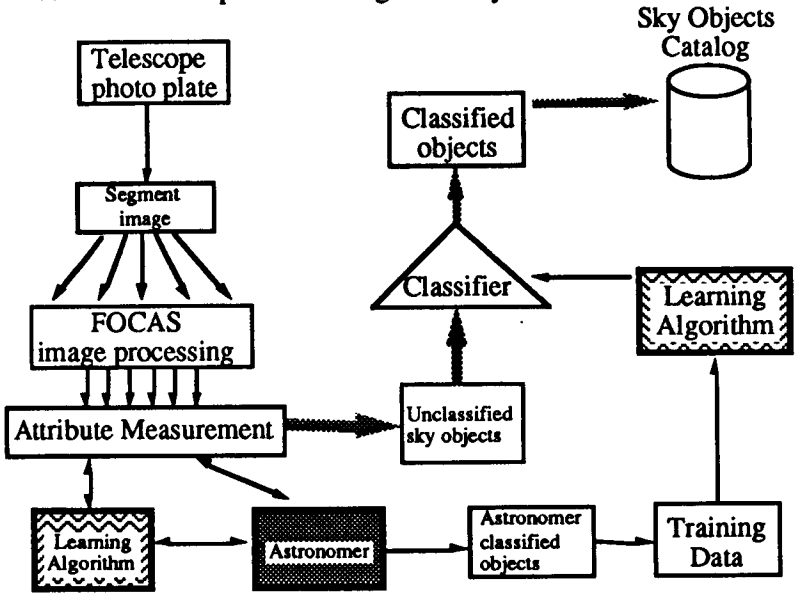


Figure 3. Architecture of the SkICAT System.

4. Classifying Sky Objects

Due to the large amounts of data being collected, a manual approach to classifying sky objects in the images is infeasible (it would require on the order of tens of man years). Existing computational methods for processing the images will preclude the identification of the majority of objects in each image since they are at levels too faint for traditional recognition algorithms or even manual inspection/analysis approaches. Our main objective is to provide an effective, objective, and examinable basis for classifying sky objects.

The photographic plates collected from the survey are being digitized at the Space Telescope Science Institute (STScI). This process will result in about 1788 digital images of roughly $23{,}000^2$ pixels each. Low-level image processing and object separation is performed by the FOCAS image processing software developed at Bell Labs [Jarv81,Vald82]. In addition to defining the objects in each image, FOCAS also produces basic attributes describing each object. A digitized plate is subdivided into a set of partially overlapping frames. Each frame represents a small part of the plate that is small enough to be manipulated and processed conveniently. Figure 3 depicts the overall architecture of the proposed SkICAT System. The discussion below will explain the loop in the bottom left-hand corner in which machine learning is employed in the attribute measurement process. The image processing steps that a digitized plate goes through are:

1. Select a frame from the digitized plate.
2. Detection: detect contiguous pixels in the image that are to be grouped as one object (standard image processing).
3. Perform more accurate local sky determination for each detected object.
4. Evaluate parameters for each object independently: we initially measured 18 *base-level attributes*.
5. Split objects that are "blended" together and re-evaluate attributes.
6. **AUTOPSF**: select a subset of the objects in the frame and designate them as being "sure-thing stars, form PSF template.
7. Measure *resolution scale* and *resolution fraction* attributes for each object: These are obtained by fitting the object to the template of sure-thing stars formed in step 6.
8. Classify objects in image.

All steps are automated except for steps 6 and 8. Step 6 needs further elaboration. The goal of this step is to define the two resolution attributes mentioned in step 7. These attributes are parameters of a template defined on a point spread function (PSF). The template is computed over a subset of objects identified as sure-thing stars. The sure-thing stars are selected by the astronomer. They represent the "archetypal" stars in that image. Once the stars are selected, the template fitting and resolution parameter measurements are computed automatically. Thus in order to automate steps 1-7 we need to automate the star selection step (6). We refer to this problem as the *star selection subproblem.*

The 18 base-level attributes measured in step 4 are [Vald82]:

- isophotal magnitude
- isophotal area
- core magnitude
- core luminosity
- sky brightness
- orientation
- sky sigma (variance)
- image moments (8): ir1, ir2, ir4, r1, r2, ixx, iyy, ixy.
- eccentricity (ellipticity)
- semi-major axis
- semi-minor axis.

Once all attributes, including the resolution attributes, for each object are measured, step 8 involves performing the final classification for the purposes of the catalog. We are currently considering classifying objects into four major categories: star (s), star with fuzz (sf), galaxy (g), and artifact (long). We may later refine the classification into more classes, however, classification into one of the four classes represents our initial goal.

4.1. CLASSIFYING FAINT OBJECTS AND THE USE OF CCD IMAGES

In addition to the scanned photographic plate, we have access to CCD images that span several small regions in some of the frames. CCD images are obtained from a separate telescope. The main advantage of a CCD image is higher resolution and signal-to-noise ratio at fainter levels. Hence, many of the objects that are too faint to be classified by inspection of a photographic plate, are easily classifiable in a CCD images. In addition to using these images for photometric calibration of the photographic plates, we make use of CCD images in two very important ways for the machine learning aspect:

1. CCD images enable us to obtain class labels for faint objects in the photographic plates.
2. CCD images provide us with the means to reliably evaluate the accuracy of the classifiers obtained from the decision tree learning algorithms.

Recall that the image processing package FOCAS provides the measurements for the base-level attributes (and the resolution attributes after star selection) for each object in the image. In order to produce a classifier that classifies faint objects correctly, the learning algorithm needs training data consisting of faint objects labeled with the appropriate class. The class label is therefore obtained by examining the CCD frames. Once trained on properly labeled objects, the learning algorithm produces a classifier that is capable of properly classifying objects based on the values of the attributes provided by FOCAS. Hence, in principle, the classifier will be able to classify objects in the photographic image that are simply too faint for an astronomer to classify by inspection. Using the class labels, the learning algorithms are basically being used to solve the more difficult problem of separating the classes in the multi-dimensional space defined by the set of attributes derived via image processing. This method is expected to allow us to classify objects that are at least one magnitude fainter than objects classified in photographic sky surveys to date.

4.2. INITIAL RESULTS FOR THE CLASSIFICATION PROBLEM

Starting with digitized frames obtained from a single digitized plate, we performed initial tests to evaluate the accuracy of the classifiers produced by the machine learning algorithms ID3, GID3*, and O-BTree. The data consisted of two frames from a single plate. The two frames were chosen such that we had a CCD counterpart for each of them. The first frame contains the Abell 68 cluster of galaxies (A68) and the second frame contains the Abell 73 cluster (A73). A68 has 88 objects and A73 has 96. We trained the algorithms on training data from one frame and then used the second frame to independently evaluate the accuracy of the decision tree produced. The results may be summarized as follows: Algorithms GID3* and O-BTree produced significantly better trees than ID3. Accuracy for GID3* was about 90%. O-BTree's accuracy was slightly better and the trees generated by O-BTree were on average more compact (smaller number of leaves). O-BTree produced trees that on average had about 6 leaves.

In contrast, the best ID3 tree had 10 leaves and an error rate of 20.5%.

However, when the same experiments were conducted without using the *resolution scale* and *resolution fraction* attributes of step 6, the results were significantly worse. The error rates jumped above 20% for O-BTree, above 25% for GID3*, and above 30% for ID3. The respective sizes of the trees grew as well.

The initial results may be summarized as follows:

1. Algorithms GID3* and O-BTree produced significantly better trees than ID3.
2. Classification accuracy results of better than 90% were obtained when using two user-defined attributes: *resolution fraction* and *resolution scale.*
3. Classification results were not as reliable and stable if we exclude the two resolution attributes.

We took this as evidence that the resolution attributes are very important for the classification task. Hence we turned to addressing the star selection subproblem in order to automate step 6 above. Furthermore, the results point out that the GID3* and O-BTree learning algorithms are more appropriate than ID3 for the final classification task.

4.3. AUTOMATING THE STAR SELECTION PROCESS

Based on the initial results of the previous section, it was determined that using the resolution attributes is necessary since without them the error rates were significantly worse. We do not have the option of leaving star selection as a manual step in the process, since it is a time consuming task and will easily become the bottleneck in the system. We decided to use a machine learning approach to solve the star selection subproblem.

The star selection subproblem is a binary classification problem. Given a set of objects in an image, the goal is to classify them as sure-thing stars and non-sure-thing stars. Using the data from A68 and A73 we were able to obtain better than 94% accuracy on selecting stars. However, these data sets came from a single plate. We also needed to evaluate the robustness of the produced classifiers when going across plates: i.e. test the classifier on images from plates other than the plate from which the training data was drawn. Since we had access to only one plate scanned by STScI, we decided to use plates scanned by a lower resolution scanner: the COSMOS scanner at the Royal Observatory, Edinburgh (ROE). We obtained frames from three different POSS-II plates: F2268, F2249, and F830.

Although our cataloguing task will eventually use STScI scans, we decided to use the COSMOS scans to conduct our initial testing. It is strongly believed that the results obtained on the COSMOS scan will be lower bounds on the performance attainable on the higher quality STScI scans. We constructed training data using subsamples of F2268 and F2249. Data from F830 were to be used strictly for testing purposes.

The data objects from all three plates were classified manually by one of the authors (N. Weir) into *sure-stars*, *non-sure-stars*, and *unknowns*. The goal of the learning subproblem is to construct classifiers for selecting out sure-stars from any collection of sky objects.

Although our accuracy on classifying data from the same plate was around 94%, the accuracy dropped to 60%-80% levels when classifying data from different plates. It was determined that the base-level attributes such as area, background-sky-levels, and average intensity are image-dependent as well as object-dependent. It was also determined that a new set of user-defined attributes needed to be formulated. These attributes were to be computed automatically from the data, and are defined such that their values would be normalized across images and plates. It is beyond the scope of this paper to give the detailed definitions of these new attributes.

4.4. CROSS-PLATE ROBUSTNESS & COMPARISON WITH NEURAL NETS

As expected, defining the new "normalized" attributes raised our performance on both intra- and inter-plate classification to acceptable levels varying between 92% and 98% accuracy. We expect the results to be better for higher resolution STScI scans, but we have not yet verified this.

In order to compare against other learning algorithms, and to preclude the possibility that a decision tree based approach is imposing *a priori* limitations on the achievable classification levels, we tested several neural network algorithms for comparison. The results indicate that neural network algorithms achieve

similar, and sometimes worse performance than the decision trees. The neural net learning algorithms tested were: traditional backpropagation, conjugate gradient optimization, and variable metric optimization. The latter two are training algorithms that work in batch mode and use standard numerical optimization techniques in changing the network weights. Their main advantage over backpropagation is the significant speed-up in training time.

Upon examining the results of the empirical evaluation, we concluded that the neural net approach did not offer any clear advantages over the decision tree based learning algorithm. Although neural networks, with extensive training and several training restarts with different initial weights to avoid local minima, could match the performance of the decision tree classifier, the decision tree approach still holds several major advantages. The most important is that the tree is easy for domain experts to understand. In addition, unlike neural network learning algorithms, the decision tree learning algorithms GID3* and O-BTree do not require the specification of parameters such as the size of the neural net, the number of hidden layers, and random trials with different initial weight settings. Also, the required training time is orders of magnitude faster than the training time required for a neural network approach.

4.5. VERIFICATION AND RELIABILITY ESTIMATES

As mentioned earlier, in addition to using the CCD frames to derive training data for the machine learning algorithms, we also use them to verify and estimate the performance of our classification technique. This is done by testing on data sets that are drawn independently from the training data. An additional source of internal consistency checks comes from the fact that the plates, and the frames within each plate are partially overlapping. Hence, objects inside the overlapping regions will be classified in more than one context. By measuring the rate of conflicting classifications, we can obtain further estimates of the statistical confidence in the accuracy of our classifier. For the purposes of the final catalog production, a method needs to be designed for resolving conflicts on objects within regions of overlap. We have not yet addressed this question.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we gave a brief overview of the use of machine learning techniques for automatically producing classification decision trees. We motivated the problem addressed by machine classification learning in a general context as well as in the particular context of our application domain: the automation of sky object catalog generation. If successful, the SkICAT system is expected to speed up catalog generation by one to two orders of magnitude over traditional manual approaches to cataloguing. This should significantly reduce the cost of cataloguing survey images by the equivalent of tens of astronomer man-years. In addition, we aim to classify objects that are at least one magnitude fainter than objects catalogued in previous surveys. Finally, this project represents a step towards the development of an objective, reliable automated sky object classification method.

The initial results of our effort to automate sky object classification in order to automatically reduce the images produced by POSS-II to sky catalogs are very encouraging. With the use of the resolution attributes we expect to have an accuracy at or above 90%. Since measurement of the resolution attributes requires interaction with the user in selecting sure-thing stars for template fitting, we used the same machine learning approach to automate the star selection process. By defining additional "normalized" image-independent attributes, we were able to obtain high accuracy classifiers for star selection within and across photographic plates. This in turn allows us to automate the computation of the powerful resolution attributes for each object in an image. This is taken as a strong indication that our automated cataloguing scheme will ultimately achieve the desired accuracy levels.

The positive initial results obtained for the star selection subproblem suggest pursuing the derivation of more image-independent attributes to describe the sky objects in an image. This is expected to lead to higher levels of classification accuracy as well as more robust classifiers. In addition, we plan to extend the basic decision tree approach we are using to one that is based on learning statistically robust rules. This approach would be based on generating multiple decision trees and selecting the best rules out of each tree. The rules are eventually merged to obtain optimal coverage of the training data sets. In our experience with the decision tree algorithms, we noticed that the decision tree produced by the learning algorithm typically has leaves which represent overspecialized or incorrect classification rules. This suggests that an overall better classifier can be constructed by generating multiple trees and selecting only good rules from each. We have adopted this methodology in the past in other domains and it has been our experience that more compact and more robust classifiers are obtained [Chen90]. We expect that adopting it here will further improve performance.

Final object classification will be, to some extent, also a matter of scientific choice. While objects in every catalog will contain a classification entry, all of the object attributes will be recorded as well. One could therefore reclassify any portion of the survey using alternative criteria better suited to a particular scientific goal (e.g. star catalogs vs. galaxy catalogs). The catalogs will also accommodate additional attribute entries, in the event other pixel-based measurements are deemed necessary. An important feature of the survey analysis system will be to facilitate such detailed interactions with the catalogs.

As part of our plans for the future we also plan to begin investigation of the applicability of unsupervised learning (clustering) techniques such as AUTOCLASS [Chee88] to the problem of discovering clusters or groupings of interesting objects. The goal is to evaluate such a capability as an aid for the types of analyses astronomers conduct after objects have been classified into known classes. Typically, astronomers examine the various distributions of different types of objects to test existing models of the formation of large-scale structure in the universe. Armed with prior knowledge about properties of interesting clusters of sky objects, a clustering system can search through catalog entries and point out potentially interesting object clusters to astronomers. This will help astronomers catch important patterns in the data that may otherwise go unnoticed due to the sheer size of the data volumes.

ACKNOWLEDGMENTS

We would like to thank the COSMOS group at the ROE, in particular H. T. MacGillivray for providing the scan data for this work. Also, many thanks to the Sky Survey team for their expertise and effort in acquiring the plate material. The POSS-II is funded by grants from the Eastman Kodak Company, The National Geographic Society, The Samuel Oschin Foundation, NSF Grants AST 84-08225 and AST 87-19465, and NASA Grants NGL 05002140 and NAGW 1710. We thank Joe Roden for help on evaluating the performance of the learning algorithms. This work was supported in part by a NSF graduate fellowship (N. Weir), the Caltech President's Fund, NASA contract NAS5-31348 (S. Djorgovski and N. Weir), and the NSF PYI Award AST-9157412 (S. Djorgovski).

The work described in this paper was carried out in part by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[Brei84] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees.* Monterey, CA: Wadsworth & Brooks.

[Chee88] Cheeseman, P., Self, M., Kelly, J., Taylor, W. Freeman, D. and Stutz, J. (1988) "Bayesian Classification." *Proceedings of the Seventh National Conference on Artificial Intelligence AAAI-88*, pp. 607-6611, Saint Paul, MN.

[Chen90] Cheng, J., Fayyad, U.M., Irani, K.B. and Qian, Z. (1990) "Applications of machine learning techniques in semiconductor manufacturing." *Proceedings of the SPIE Conference on Applications of Artificial Intelligence VIII.* pp. 956-965, Orlando, Fl.

[Feig81] Feigenbaum, E.A. (1981) "Expert systems in the 1980s." In Bond, A. (Ed.) *State of The Art Report on Machine Intelligence.* Maidenhead: Pergamon-Infotech.

[Fayy90] Fayyad, U.M. and Irani, K.B. (1990). "What should be minimized in a decision tree?" *Proceedings of Eighth National Conference on Artificial Intelligence AAAI-90*, Boston, MA.

[Fayy91] Fayyad, U.M. (1991). *On the Induction of Decision Trees for Multiple Concpt Learning.* PhD Dissertation, EECS Dept. The University of Michigan.

[Fayy92a] Fayyad, U.M. and Irani, K.B. (1992) "On the handling of continuous-valued attributes in decision tree generation." *Machine Learning*, vol.8, no.2.

[Fayy92b] Fayyad, U.M. and Irani, K.B. (1992) "The attribute selection problem in decision tree generation." *Proceeedings of the Tenth National Conference on Artificial Intelligence, AAAI-92.* San Jose, CA.

[Jarv81] Jarvis, J. and Tyson, A. (1981) *Astronomical Journal* 86:41.

[Quin86] Quinlan, J.R. (1986) "The induction of decision trees." *Machine Learning* vol. 1, no. 1.

[Vald82] Valdes (1982) *Instrumentation in Astronomy IV*, SPIE vol. 331, no. 465.

INTELLIGENT ASSISTANCE IN SCIENTIFIC DATA PREPARATION

Steve Chien, R. Kirk Kandt, Joseph Roden, Richard J. Doyle, and Scott Burleigh

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099

Todd King and Steve Joy

Institute of Geophysics and Planetary Physics
University of California at Los Angeles
Los Angeles, CA 90024-1406

Abstract

Scientific data preparation is the process of extracting usable scientific data from raw instrument data. This task involves noise detection (and subsequent noise classification and flagging or removal), extracting data from compressed forms, and construction of derivative or aggregate data (e.g. spectral densities or running averages).

A software system called PIPE provides intelligent assistance to users developing scientific data preparation plans using a programming language called Master Plumber. PIPE provides this assistance capability by using a process description to create a dependency model of the scientific data preparation plan. This dependency model can then be used to verify syntactic and semantic constraints on processing steps to perform limited plan validation. PIPE also provides capabilities for using this model to assist in debugging faulty data preparation plans. In this case, the process model is used to focus the developer's attention upon those processing steps and data elements that were used in computing the faulty output values. Finally, the dependency model of a plan can be used to perform plan optimization and runtime estimation. These capabilities allow scientists to spend less time developing data preparation procedures and more time on scientific analysis tasks.

Introduction

Scientific data preparation is defined as the application of multiple transformations to collected data sets in order to produce data in an easily usable form. The questions a scientist asks dictate which data are to be collected as well as which transformations are to be applied. The need for simplified scientific data preparation has increased due to the volume of data now collected and the diverse uses for any specific type of data. Automated scientific data processing systems can be used to simplify this process.

While general scientific data processing systems have existed for some time, the complexity of data types and transformations required in specific domains renders these systems of limited utility. As a result, many scientific teams develop their own software systems to accomplish the data preparation required in their specific domain. These systems suffer because they become too specific, and the effort spent developing such systems are only of value within the context of a particular domain and task. Because scientists desire to reuse their work, hybrid systems are appearing which provide useful analysis tools and definition of domain-specific data types and transformations. Plans are developed in these systems which specify which of the transformations to apply to a collection of data sets. By the nature of the processing steps required in many domains, these plans can become quite complex. We are now at a point where the complexity of these tools requires significant expert knowledge to use.

Master Plumber [King & Walker 1991] is a software tool developed by the UCLA Institute of Geophysics and Planetary Physics to create programs to prepare scientific data. While its primary area of application has been time-series magnetometer data, the tool is applicable to the general task of scientific data preparation.

Master Plumber is a dataflow system. Thus, in Master Plumber, data elements are represented by columns, which are streams of data being processed as they move through the system. Data processing steps are called fittings, and a plan to process a particular form of a dataset into another form is called a blueprint.

Thus, as shown in Figure 1, raw data might be read in using an intro_flatfile fitting, a running average computed using a runstat fitting, and the results written into an output file.

```
1.  intro_flatfile infile=foo
      columns=bx
```

```
2.  runstat length=1287 shift=1
      columns=bx
3.  write_flatfile outfile=bar
      columns=bx,rabx overwrite=YES
```

Figure 1: A Simple Blueprint

A major difficulty in constructing blueprints is tracking the many fitting and column interactions. While a typical blueprint might use 25 columns and 20 fittings, the more complex blueprints use hundreds of columns and 30 or more fittings. Because of the number of possible interactions, constructing and debugging scientific data preparation blueprints is a time-consuming task requiring expert knowledge.

Because of the complexity of the data preparation task, users sometimes make errors in blueprint construction. One type of construction error occurs when a user forgets to set up the data needed for a particular step. Unfortunately, this type of error can go unnoticed until far into the execution of the blueprint, wasting valuable time.

Another common situation is that the exact method of processing the data is dependent upon the character of the data. In this case the user will use some default methods for processing the data, examine the results, and modify the options. This tuning cycle continues until the data is in a satisfactory form.

The final aspect of blueprint development which complicates the development process is that new fittings are added to a system as new needs and requirements arise. In addition, new fittings also evolve with new options and characteristics being added. Any intelligent tool must be readily changed to remain useful in such a dynamic environment.

Currently there are approximately 65 fittings which are part of the standard Master Plumber system. These fittings perform a variety of transformations on the data flow, such as: introducing and writing data into several formats; displaying data on the screen; and actual numerical transformations. There are support libraries which allow for fittings to be written in either C or FORTRAN. A special fitting called PLISP takes programs written in a C-like language and performs the transformations on the data flow. This allows for new processing steps to be initially tested as PLISP programs and later be integrated as full-fledged fittings into the Master Plumber system.

Some scientists use data preparation systems indirectly with the help of software support personnel who write and debug the actual data preparation plans. The goal of PIPE is to make Master Plumber easy enough to use such that this type of support is not necessary. The combination of PIPE and Master Plumber will allow the blueprint developer to develop blueprints easier and faster, allowing them to spend more time on data analysis and less time on data preparation.

Overview

To achieve these goals of assistance in the scientific data preparation process, PIPE [Chien et al. 1992] provides four capabilities:

1. constraint checking to detect invalid blueprints before execution;
2. diagnosis assistance of blueprints through dependency analysis;
3. optimization of blueprints through dependency analysis; and
4. runtime estimation, using models of fitting runtime performance.

The architecture of the PIPE system is shown in Figure 2. PIPE accepts a blueprint file and a set of descriptors for datafiles and uses a fittings knowledge base to construct a dependency graph representing the computations to be performed by each of the fittings in the blueprint. This blueprint parsing phase uses knowledge of fittings and their options to construct a dependency graph, which indicates for each fitting which columns are accessed and used to modify existing columns, create new columns, or remove existing columns. This dependency graph can then be used by the constraint checking module which determines if any of the constraints associated with the fittings have been violated.

In cases where blueprints must be debugged, PIPE can use the dependency graph to support isolation of the fault in the blueprint. Because the dependency graph tracks all of the operations upon the columns, when the user detects an error in one of the output columns, PIPE can present a list of fittings which modified the column in question. The user can then focus his attention upon these fittings, to determine where the error was introduced into the data, sometimes by plotting intermediate data. After isolating the first fitting at which the column is faulty, the user can query PIPE for information on the fitting to determine which columns were used to compute the changed column. This process continues until the fault is isolated to the data, fitting option settings, or fitting code itself.

PIPE also provides an optimization capability. Because PIPE constructs a full computation dependency graph, PIPE can determine the last fitting in which each column of data is used in the blueprint. Thus unneeded data can be removed from the dataflow, decreasing the execution time Because many fittings operate on data by default, PIPE distinguishes between default processing and explicit processing. Default computation which does not result in a program output (e.g. plot, output file) can also be removed.

Finally, PIPE provides a runtime estimation capability. Using the dependency graph to determine which columns each fitting processes, and models of runtime for each fitting type, PIPE can provide an estimate of how long the

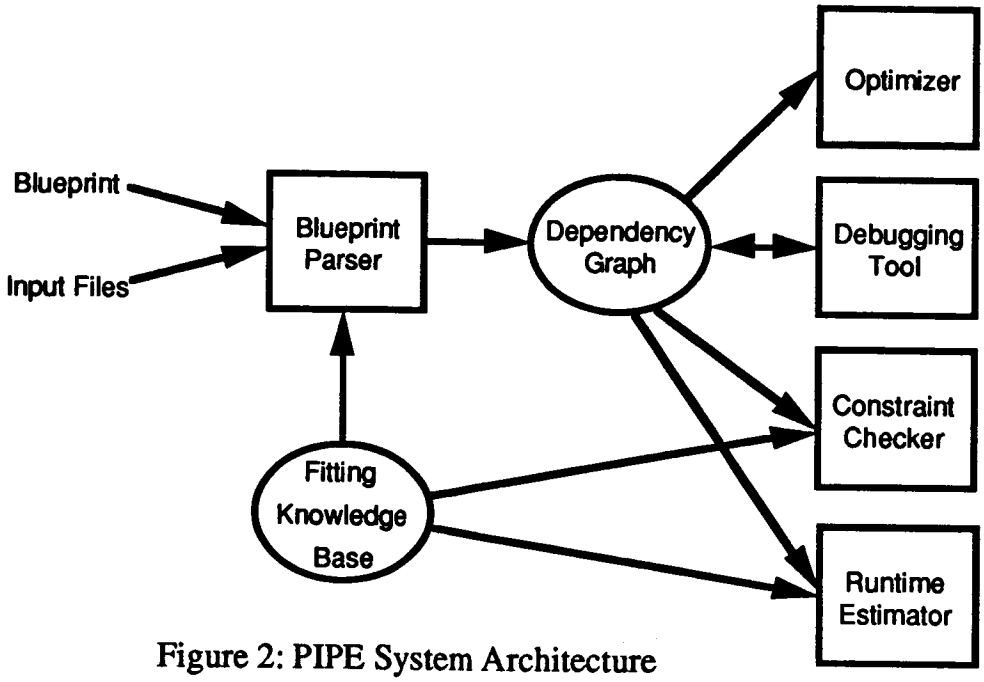


Figure 2: PIPE System Architecture

blueprint will take to run to completion for the specified datafiles.

Blueprint Parsing

In order to provide assistance in blueprint development, PIPE constructs a dependency network representation of a blueprint. When a blueprint is read in by PIPE, it is processed from the first step onward. For each fitting, PIPE uses:

- methods stored in the fitting knowledge base,
- default values stored in the fittings knowledge base,
- fitting options,
- a list of existing columns in the flow, and possibly
- an input file

to determine:

- any new columns created by the fitting,
- any existing columns modified by the fitting,
- existing columns deleted by the fittings.

Additionally, for any new or modified columns, PIPE determines:

- the set of columns accessed in computing the value for the column.

Because columns may be processed by default or explicitly selected, the dependency network also makes note of this distinction. This facet of the processing is important in order to take appropriate action when optimizing the blueprint (see below).

Constraint Checking

Constraint checking occurs while the blueprint file is being parsed (i.e., prior to execution). A description of the constraint checking algorithm follows.

```
During Parsing
   for each fitting in the blueprint
      for each option specified
         check option type constraints
      check for required options

After Parsing
   for each parsed fitting in blueprint
      for each option in fitting
         check option value constraints
      check inter-option constraints
      check dependency constraints
   check inter-fitting constraints
```

Diagnosis Assistance

PIPE also provides a blueprint diagnosis facility. This capability supports two basic types of queries: column-centered queries and fitting-centered queries. The column-centered queries are of the form

```
"What fittings affected <column>
before <fitting>?"
```

and default to the entire blueprint. This question can be easily answered using information from the dependency network. PIPE steps through the fittings in the blueprint and determines those fittings which create, modify, or delete <column>. This list of fittings is then displayed to the user in graphical form. The fitting centered queries are of the form

```
"What columns did <fitting>
affect?", and

"What columns did <fitting> access
in performing its processing to
affect these columns?"
```

These types of queries can be answered by interpreting the dependency graph information on the designated fitting. The first query can be answered by determining the set of columns created, modified or deleted by the fitting. The second query can be answered by accessing dependency network information regarding which columns were accessed by the fitting in performing these operations.

Blueprint Optimization

PIPE also provides a limited blueprint optimization capability. In this capability, PIPE examines the dependency graph of each column and determines the last fitting at which each column is accessed explicitly (i.e., not by default). PIPE then recommends removing this column immediately after this fitting. If this column is not processed in the remainder of the blueprint, this removal does not significantly alter the runtime of the blueprint. However, many of the fittings process all of the columns in the flow by default. Thus, when a column that is processed in the remainder of the blueprint is removed from the data flow a significant speedup can result. While commonly used blueprints are likely to have unused columns optimized by hand, automating this process relieves the user of the burden of determining the point at which a column can be removed. Additionally, by allowing PIPE to automatically determine the correct places to remove columns, PIPE reduces the chance that a user will inadvertently prematurely remove a column from the data flow, which would cause an error.

Runtime Estimation

The final capability that PIPE provides is runtime estimation. PIPE estimates the runtime of a blueprint for a specific data set by applying the following algorithm:

```
for each fitting in the blueprint
   identify fitting runtime model
   compute runtime given dataset size
   add runtime to total runtime
   compute new size of dataset
```

Tracking the size of a dataset in Master Plumber can be a difficult task. Original data set sizes are determined from input files. When data of different temporal granularity are introduced into an existing flow, or when decimation operations are performed, data set sizes will need to be recomputed. Sometimes a fitting can affect the size of the dataset in a manner that depends on the exact data processed. In these cases, the exact dataset size cannot be determined, so PIPE estimates the size of the dataset at the output of the fitting. These estimations are sufficient for giving the user reasonably accurate runtime estimates.

An Example

We now illustrate each of the capabilities of PIPE using example blueprints. For an example of constraint checking, suppose a user has created a blueprint containing the following statement:

```
4. bin columns=bx delta=60.0 min_max
```

Because the option `min_max` requires that a value be specified, PIPE would indicate a constraint error such as:

- `Fitting 4. bin option min_max required value not found; string type required.`

As another example of the constraint checking, consider the following blueprint statement:

```
7. crossavg except=time avgname=xavg
```

Assuming the user removed the column named `time` earlier in the data flow, PIPE would issue a constraint error indicating:

- `Fitting 7. crossavg option except undefined column time; a column with that name was deleted at fitting 4. drano.`

An example of the diagnosis capability supported by PIPE is illustrated in the following scenario. Figure 3 shows a Master Plumber blueprint file. Suppose that the user examines the output of the blueprint and determines that column `o2` is producing results that are incorrect. The user tries to determine what may have affected column `o2` by querying PIPE:

```
Q: Which fittings created or
   modified column o2?
```

```
A: Fitting 10. drano created column
   o2.
   Fitting 12. plisp modified column
   o2.
```

The user determines that the o2 column was still incorrect before fitting 12. plisp, so the user wants to determine what columns were accessed by and were used in creating o2.

```
Q: Which columns were accessed by
   fitting 10. drano in order to
   create column o2?

A: Column raraby was accessed by
   fitting 10. drano in order to
   create column o2.
```

The user then continues backtracking through the blueprint to isolate the error:

```
Q: What fittings before fitting 10.
   drano modified column raraby?

A: Fitting 9. runstat created and
   modified column raraby.
```

By using PIPE in this way, the user can focus his attention directly upon the possibly faulty fittings instead of having to examine every fitting and column.

PIPE also uses the dependency graph to optimize blueprints. Because PIPE can determine which fittings modify which columns in the blueprint, PIPE can determine the last point at which each column is needed in the blueprint. In the example blueprint shown in Figure 3, PIPE makes the following recommendations for removal:

```
never introduce column rim

remove sens_x, senx_y, sens_z and bz
   after fitting 4

remove bx, by after fitting 8

remove rabx, raby after fitting 9

remove bxc, byc, bzc, and stime
   after fitting 12
```

PIPE also provides runtime estimation capabilities. For the optimization example shown above, PIPE estimates that the non-optimized blueprint will take 11:32 +/- 1:04 to run and the optimized blueprint will take 9:58 +/- 0:58 to run.

Discussion

There are a number of interesting directions which remain open issues. First, PIPE currently assists the user by allowing the user to track the effects of processing steps. A more intelligent system would be able to analyze the data and extract features which would inform the user as to what processing steps might be useful. For example, a system could examine the data to determine the length of gaps in data and use this information to determine whether gaps in the data need to be filled. A further analysis of the data (rates of change and Fourier analysis) might indicate what types of gap filling methods might be effective. This type of automation requires that the system possess a significantly deeper understanding of the data being processed.

Another aspect of the system is modelling the goals of the processing steps in order to make suggestions about ordering processing steps. Knowledge of the interactions between various processing steps, such as decimation of the data and computation of running averages, could be used to make suggestions on re-ordering of processing steps to improve accuracy or efficiency as needs dictate.

Distributed processing of the data is also an important issue. Because the Master Plumber system operates on data from the Planetary Data System, a distributed database, when a scientist decides to generate a specific data form, there are a number of combinations of processing and data transfer which are possible. Depending upon the data processing steps desired, it may be more efficient to process some data being accessed from a remote site before transferring it to the local site. Factors such as network transfer rates, available computer resources at each site, and current user loads at each site all affect this decision as well as the actual scientific data processing steps.

The current prototype version of PIPE was completed in July 1991. It is implemented in CommonLISP and LISPView and runs on Sun workstations.

The C++ operational version of PIPE was completed in May of 1992. and is integrated with Master Plumber and MPTool and is in use by IGPP personnel at UCLA. This version of PIPE incorporated feedback upon the "look and feel" of the interface specified by IGPP personnel.

There are numerous related projects in providing intelligent assistance in scientific computing. The Kineticist's workbench project at MIT [Abelson et al. 1989] targets modelling and analysis of dynamic systems. The SINAPSE system [Kant et al. 1990] assists in construction of numerical models for data interpretation but is specific to seismic models represented as finite difference equations. The Reason system [Atwood et al. 1990] supports analysis of high energy physics data (and is a dataflow system). Finally, the Scientific Modeling Assistant project [Keller 1991] addresses support to facilitate development of scientific models.

Summary

This paper has described a system to assist in the development of scientific data preparation programs and discussed issues in design for maintainability. This issue of maintainability was particularly important because the processing modules (fittings) are constantly evolving due to changing scientists' needs. In order to maximize

maintainability of the constraint knowledge base, information for each fitting is encapsulated in a fitting knowledge base file and as much as is practical, constraint information is represented in a general declarative fashion.

Acknowledgements

This work was performed by the Jet Propulsion laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[Abelson et al. 1989] H. Abelson, M. Eisenberg, M. Halfant, J. Katzenelson, E. Sacks, G. Sussman, J. Wisdom, and K. Yip, "Intelligence in Scientific Computing", Comm. ACM, 32(5):546-562, May 1989.

[Atwood et al. 1990] W. Atwood, R. Blankenbecler, P. F. Kunz, B. Mours & A. Weir, "The Reason Project", Stanford Linear Accelerator Technical Report #SLAC-PUB-5242, April 1990.

[Chien et al. 1992] S. Chien, R. K. Kandt, R. Doyle, J. Roden, T. King, and S. Joy, "PIPE: An Intelligent Scientific Data Preparation Assistant", Proceedings of the International Space Year Conference on Earth and Space Science Information Systems, Pasadena, CA, February 1992.

[Kant et al. 1990] E. Kant, F. Daube, W. MacGregor, J. Wald, "Synthesis of Mathematical Modeling Programs", Schlumberger Laboratory for Computer Science Technical Report Number TR-90-6, February 1990.

[Keller 1991] R. Keller, "Building the Scientific Modeling Assistant: An Interactive Environment for Specialized Software Design", Technical Report FIA-91-13, NASA Ames Research Center, Moffett, Field, CA, May 1991.

[King & Walker 1991] T. King and R. Walker, "The UCLA Data Flow System," Technical Report #3522, Institute of Geophysics and Planetary Physics, University of California at Los Angeles, CA 1991.

Session A6: MISSION OPERATIONS

Session Chair: Dr. Silvano Colombano

Process Control and Recovery in the Link Monitor and Control Operator Assistant

*Lorrine Lee
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
MS 525-3660
(818)-306-6178

Randall W. Hill, Jr.
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
MS 525-3651
(818)-306-6085

Abstract

This paper describes our approach to providing process control and recovery functions in the Link Monitor and Control Operator Assistant (LMCOA). The focus of the LMCOA is to provide semi-automated monitor and control to support station operations in the Deep Space Network. The LMCOA will be demonstrated with precalibration operations for Very Long Baseline Interferometry on a 70-meter antenna. Precalibration, the task of setting up the equipment to support a communications link with a spacecraft, is a manual, time consuming and error-prone process.

One problem with the current system is that it does not provide explicit feedback about the effects of control actions. The LMCOA uses a Temporal Dependency Network (TDN) to represent an end-to-end sequence of operational procedures and a Situation Manager (SM) module to provide process control, diagnosis and recovery functions. The TDN is a directed network representing precedence, parallelism, precondition and postcondition constraints. The SM maintains an internal model of the expected and actual states of the subsystems in order to determine if each control action executed successfully and to provide feedback to the user.

The LMCOA is implemented on a NeXT workstation using Objective C, Interface Builder and the C Language Integrated Production System.

Introduction

This paper describes our approach to providing process control and recovery functions in the Link Monitor and Control Operator Assistant (LMCOA). The focus of the LMCOA is to provide semi-automated monitor and control to support station operations for the Deep Space Network (DSN). The DSN is a world-wide network of antennas located in California, Spain, and Australia and is used for communicating with spacecraft. The antennas range from 26 to 70 meters in diameter. The 70-meter antennas are heavily oversubscribed because they are used to support communications for a large number of deep space missions. One way to increase the availability of the 70-meter antennas is to decrease the amount of time spent on overhead operations. Overhead operations are activities where the antenna is not used to support a mission operation. An example of an overhead activity is *precalibration*, which is the task of setting up the equipment at an antenna complex in order to support a communications link with the spacecraft. The process of establishing the link involves issuing control actions, or *directives*, to configure equipment at an antenna complex.

Currently, performing precalibration takes between 30 minutes to over two hours. The time varies according to operators' experience and the complexity of the operation. An example of a complex operation, which most operators don't have experience with, is Very Long Baseline Interferometry (VLBI) Delta Differential One-Way Ranging (DOR), a technique used to determine an accurate measurement of the

position of the spacecraft for navigation purposes. The actual task of performing VLBI measurements takes approximately 15 minutes, whereas precalibration for VLBI takes two to eight times as long. To address this problem, the demonstration domain for the LMCOA was selected to be precalibration for VLBI on a 70-meter antenna.

Precalibration is currently a time consuming process because of limitations in the existing operational monitor and control system. It is a command-line, keyboard-entry system that requires operators to manually send hundreds of directives to subsystems and monitor over a thousand incoming messages on a text-based scrolling log. The system lacks explicit, informative responses about the state of a directive and does not provide guaranteed communications between the monitor and control system and subsystems being controlled. For each directive sent by the operator, the subsystem returns a *directive response* which is simply an acknowledgment from the subsystem that the directive was received. A directive response does not indicate the successful or unsuccessful execution of a directive. The subsystem may also send out *event notice messages*, which relay information about the state of some device in a subsystem. However, these messages are not explicitly tied to any directive that was sent. The operator must rely on his/her experience to determine which directive was most likely to have caused the subsystem to send the event notice message. *Monitor data*, which are sent periodically by the subsystems, also provides information about device states. However, monitor data is never displayed automatically or tied to any directive. Instead, a subset of monitor data is formatted into pre-defined displays that the operator can invoke. The operator has to decide which piece of the data (s)he's interested in and which display contains that piece of information before (s)he can access it.

The absence of guaranteed communications between the existing monitor and control system and the subsystems cause false alarm situations when communication packets destined for the monitor and control system are dropped. The subsystem cannot resend dropped packets because it can't detect them. The number of false alarms inundates the user and hides real alarm situations. Finally, the system is prone to input errors. A simple precalibration pass requires over 200 directives to be issued. The operator must manually identify and type each directive and its parameters. Simple typographic errors can cause a subsystem to take several minutes to recover.

In the existing system, the burden of filtering through the data returned by the subsystems and determining whether execution is proceeding as expected rests completely on the operator. By providing semi-automated control tools and improved data presentation, the LMCOA lightens the operators' load and frees them to do tasks that require human judgment and actions. The overall benefits of the LMCOA are increased DSN resource availability and operator efficiency. The LMCOA prototype will demonstrate the following key features: 1) closed loop control, which provides the operator with explicit and consistent feedback about the executing state of directives; 2) anomaly detection and explanation for directive rejections and anomalous device states; 3) recovery assistance by suggesting alternate plans of action; and 4) a consistent graphical user interface with multi-level displays. The prototype will be demonstrated within the framework of the existing monitor and control system using only the information that is available to the current system.

Temporal Dependency Network

The current system does not have any single source of documentation that outlines operational procedures for VLBI. Rather, it has many operations manuals which are sometimes inconsistent, out-of-date and difficult to use. Therefore, actual operations rely heavily on operator experience and expertise. Given this situation, our first task was to learn the procedures and lay them out in a coherent, consistent manner using a Temporal Dependency Network (TDN).

A TDN, shown in Figure 1, is a directed graph of interconnected nodes that represents an end-to-end operational sequence for VLBI Delta DOR. Sequential, parallel, and optional operation sequences are identified in the TDN. Each node in the TDN contains a block of directives that are sent to the subsystems sequentially. Nodes have precedence constraints where the block of directives in a node cannot be sent until all of its predecessor nodes' directives have successfully completed execution. Each node has associated precondition and postcondition constraints. These constraints define the state the system must be in before the start of each block of directives and after successful execution of those directives. Each node may also have time constraints which limit the start and completion of the directives to a specific time or time interval.

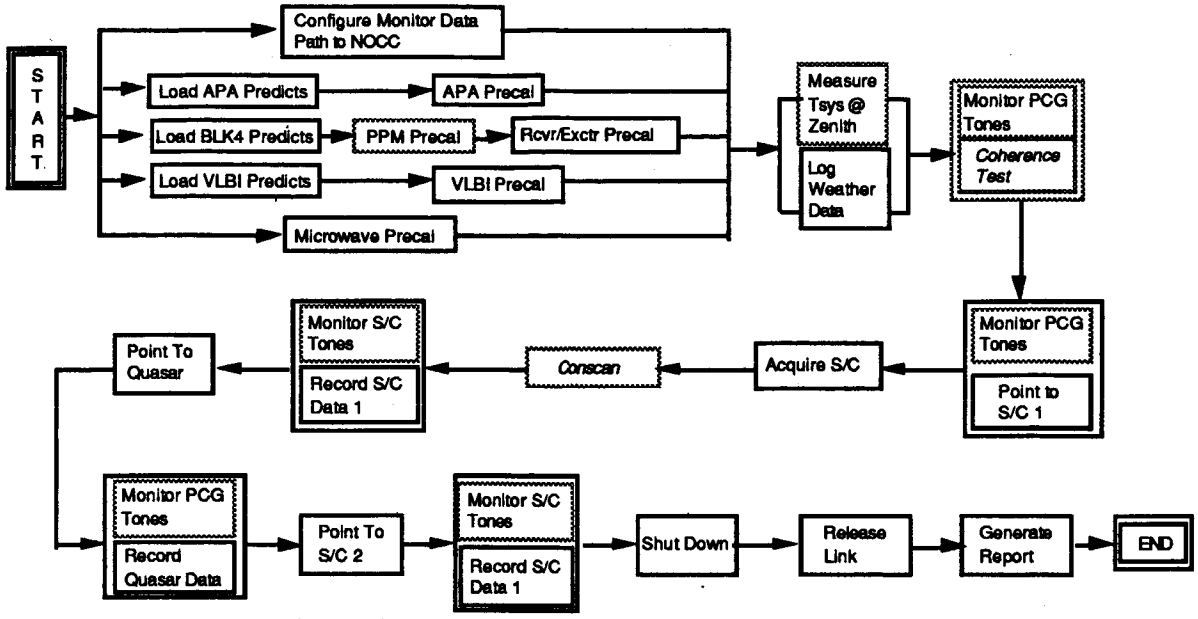


Figure 1. VLBI Temporal Dependency Network.

Architecture

Our goal is to provide both closed-loop control and closed-loop communications for the operator. There are two major modules in the LMCOA which control the TDN execution and close the communications and control loops. These are the TDN Execution Manager and the Situation Manager (SM). Other modules which will be discussed are the Block Execution module, Router, Timed-Events Manager, Monitor Data Handler and DSN Data Simulator. An overview of the architecture is presented in Figure 2.

The LMCOA is an UNIX based, object-oriented, multi-*threaded* architecture. A thread is a very lightweight process that only carries the basic information about the processor state, such as a program counter and hardware registers, that is necessary for independent execution. Many independently executing threads can make up one program. In a single processor system, threads are useful for providing concurrent logical control. The modules in the LMCOA are implemented as individually executing threads that cooperate through message passing and manipulation of shared data objects.

TDN Execution Manager and Block Execution Modules

The TDN execution manager is responsible for traversing the TDN in order to identify blocks and directives that are ready to execute. The execution manager identifies the blocks whose precedence constraints, as laid out in the TDN, are satisfied. (Blocks and nodes will be used interchangeably since a node consists of a block of directives.) Each block that is identified as having its precedence constraints satisfied is executed concurrently as separate instances (or threads) of the block execution module. Multiple block execution threads may be active at the same time because of parallelism in the TDN. By executing each block as a separate thread, we allow the operating system to schedule execution of parallel blocks on a single processor system. A block that is waiting for some event (e.g. a subsystem's response to a directive) does not hold up any other blocks except for its own successor blocks.

When a block first starts execution, it asks the SM to evaluate its block preconditions. A block's directives are sent only after the SM verifies that the preconditions are satisfied. After a directive is sent, a directive response must be received before the next directive in the block is sent to a subsystem. After the last directive is sent and its corresponding response is received, the block execution module asks the SM to evaluate the block's postconditions. If the postconditions are satisfied, the block of directives is considered

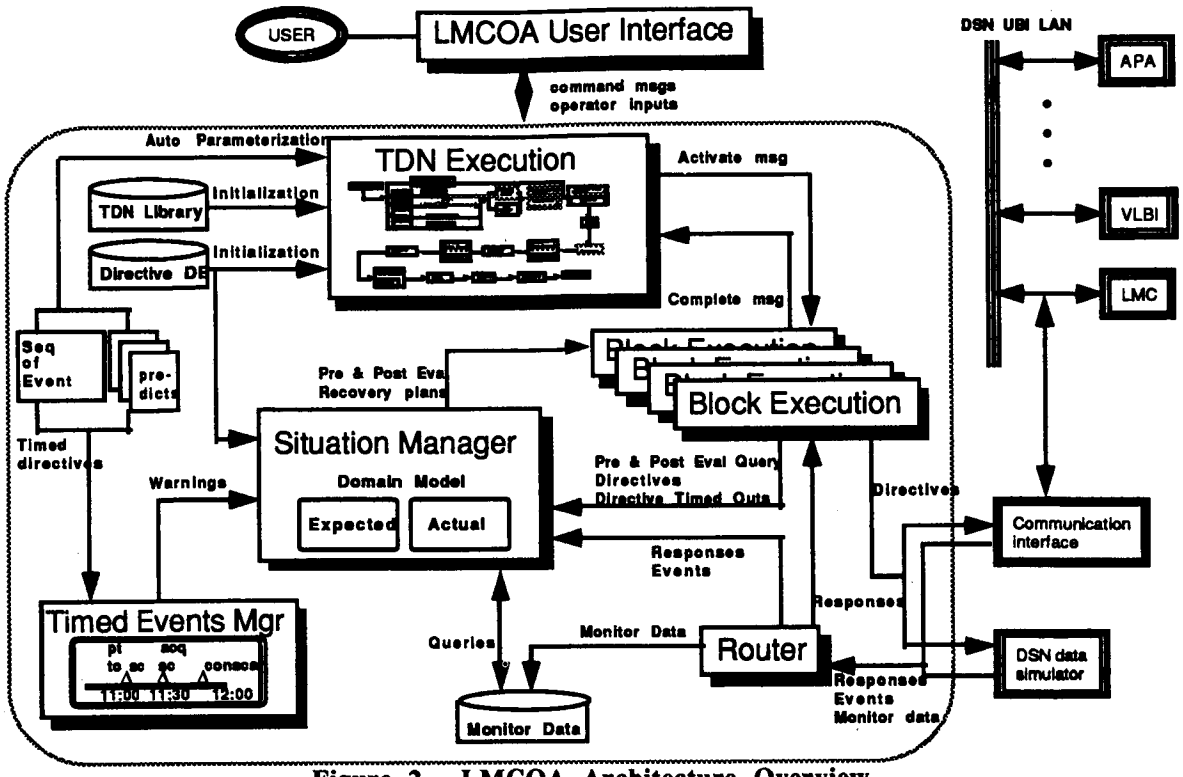


Figure 2. LMCOA Architecture Overview

completed. For each completed block, the TDN execution manager identifies all successor blocks. Each successor block that satisfies its precedence constraints is forked as a separate thread. In this manner, the TDN is traversed and all of the directives in the TDN blocks are sent to the subsystems.

Situation Manager Device Model

The Situation Manager (SM) is an AI-based module that controls the start and completion of each block of directives, verifies correct execution by checking postcondition constraints on a block of directives, detects problems and provides recovery assistance. The goal of the SM is to maintain situational awareness. In order to accomplish this, the SM keeps an internal model of all the monitorable hardware and software devices in the system. Each device represented in the model has attributes that reflect the state of the device. Each attribute has a pair of values: an *expected value* and an *actual value*. The expected value of an attribute is set when a directive is sent to the subsystem. The actual value of an attribute is set when the subsystem sends messages noting state changes in the subsystem.

Every directive sent to a subsystem is expected to effect certain known changes on the states of the devices in the subsystem. In the SM, each directive has a production rule associated with it which sets the expected states of the affected device attributes to the values expected if the directive executes successfully. Thus each time a directive is sent, the expected values of the attributes in the device model are updated. Several data sources are used to set the actual values of the device attributes: event notice messages, monitor data and the operator. All three provide information about the actual state of a device but no explicit information about whether a directive was executed successfully. However, by using information about the expected and actual states of devices, the success of a directive can be determined.

Event notice messages explicitly give the actual states of devices. Since the numbers and types of event notice messages are finite and known, we know in advance which device attributes the message refers to. Therefore, for each event notice message, the SM has a production rule that updates the actual values of the relevant device attributes.

Monitor data are blocks of status information that are sent periodically by the subsystems. Monitor data usually provides more information than event notice messages. Since monitor data is sent only at set intervals by a subsystem, the information is stored in a Monitor Data Database. When the SM needs monitor data to verify the state of a device, it queries the Monitor Data Database for the latest value. The time of the returned monitor data value is checked and if it is more recent then the time at which the directive was sent, this value is used as the actual value.

Finally, the operator is provided a set of pre-defined monitor displays. The information in these displays is not always available from the monitor data blocks. These displays are generated as bit-map displays at the subsystem level and are unavailable to the LMCOA because of format and DSN operational restrictions. Therefore, for certain directives, the operator must obtain information from the displays and enter it into the LMCOA. This information is used to set the actual value of an attribute in the SM internal device model.

Because the kinds of information available in event notice messages and monitor data are known, we can associate with each device attribute the data source which is used to set its actual state. The primary data source is always event notice messages. If there are no event notice messages, the secondary data source is the Monitor Data Database. Finally, if monitor data doesn't provide the necessary information, the final data source is the operator.

Process Control, Anomaly Detection and Recovery

The SM provides control over the start and completion of each block in the TDN. A TDN block must wait for a message from the SM indicating preconditions are satisfied before it can send the first directive. Likewise, a TDN block cannot mark itself as completed until it receives a message from the SM indicating postconditions are satisfied. Once that message is received, the successor blocks of the just completed block can start execution. The SM also controls continuation of the TDN block after an anomaly is detected. It detects the anomaly and constructs a recovery plan to insert into the TDN. There are several methods for detecting anomalies during directive execution: evaluating directive responses and event notice messages, comparing expected and actual values in the device model, and evaluating precondition and postcondition constraints.

The subsystems send several types of directive responses: PROCESSING, COMPLETED, and REJECTED. The PROCESSING directive response means that the directive was received but has not started executing yet. The COMPLETE directive response means that the directive was received and is currently being executed, and when the subsystem cannot interpret a directive, a REJECTED response is received. Except for a few known exceptions, the subsystems will respond to every directive sent with either a COMPLETE or a REJECTED response. In addition, some subsystems may precede the COMPLETE response with a PROCESSING response. The REJECTED response signals an incorrectly formatted or parameterized directive. Again, the numbers and types of REJECT responses are known and recovery is based on existing plans documented in the operations manual. Each REJECT response has a rule associated with it which selects the appropriate recovery plan.

To address the problem of dropped communication packets, the LMCOA sets a time limit on when a response must be received. If the response is not received within a specific interval, recovery is started. Recovery from a timed out directive involves querying the Monitor Data Database or the operator for the actual value of a device. If the actual does not match the expected, it is an indication that the directive was not received and the directive is resent.

Event notice messages provide information about the state of a device including whether or not the device is in an erroneous state. Each time an event notice message is used to update an actual value of an attribute, it is compared to the expected value. If the actual and expected values do not match, then the system is in an erroneous state. Associated with each event notice message that describes an erroneous state is a rule that selects a known recovery plan. Precondition and postcondition evaluations are handled the same way. The actual values of the device states are compared to those defined by the preconditions or postconditions. If the states do not match, then a recovery plan to correct the state is created.

Once a recovery plan is created, it must be integrated into the TDN in order to be executed. A recovery subnet containing blocks of directives, control logic and display directives is created. This subnet carries

the same features as the TDN such as time, precedence, precondition and postcondition constraints and allows representation of parallel and optional sequences. It is inserted immediately after the anomalous block and becomes an integral part of the TDN. Recovery directives are sent and monitored in the same way as any other directive in the TDN.

The communications loop for a directive is closed when the TDN block execution module receives a directive response or completes recovery from a reject message. The control loop for a directive is closed when its block postconditions are satisfied and any recovery subnets are completed. The operator is thus guaranteed appropriate action will be taken for every directive sent.

Router, Timed-Events Manager, Monitor Data Handler and DSN Subsystem Simulator

In addition to the TDN execution, block execution and SM modules, there are several other supporting modules. The Router is responsible for all communications between the LMCOA and the DSN subsystems. It receives and decodes input from the DSN subsystems and directs the input to either the TDN Execution Manager or the SM or both. It also formats the directives into communication packages that are sent to the subsystems. The Timed-Events Manager maintains a list of directives in the TDN with time constraints. Two warning periods are associated with each of these timed directives. As the time nears the first warning period, the operator is advised to take an action such as skipping optional sequences. As the time nears the second warning period, the SM executes a known recovery plan to skip any optional sequences still pending. The Monitor Data Handler receives Monitor Data blocks from the subsystems and stores them in the Monitor Data Database. It also handles queries from the SM and returns the latest value for the requested monitor data element. Because access to the operational environment is limited, a DSN Subsystem Simulator was implemented to simulate the directive responses and event notice messages from the subsystems.

Automated parameterization and Automated Report Generation

One of our goals is to decrease the amount of keyboard entry required of the operator. In the existing system, the operator looks at several sets of paper listings to identify critical directive parameters. In the LMCOA, these listings are accessed in electronic form, parsed, and used to parameterize the directives defined in the TDN. In the existing system, the operator jots down the time at which certain directives were executed and their results. At the end of the pass, the operator writes a set of paper reports. In the LMCOA, we internally log the time, parameters and responses for each directive and the reports are automatically generated at the end of the pass.

User Interaction and Status Displays

Another one of our goals is to provide consistent interaction and meaningful displays to keep the user aware of what's going on in the system. The primary method of interaction is via menu or button selections with a mouse. Occasionally the operator may be asked to enter a value or response. The primary interaction window is a block-level display of the TDN which provides a high-level end-to-end sequence of operations. A color bar in each block is used to show the status and progress of each block: a gray bar means the block is inactive, a green bar means the directives are executing, a red bar means an anomaly has occurred and a blue bar means the directives have successfully completed. The portion of the color bar that is green is proportional to the number of executed directives in the block.

The operator can bring up a lower level display for each block that shows the state of the block, the state of each directive in the block (inactive, executing, paused, anomaly, etc.) and lists the preconditions and postconditions for each block. There are TDN-level controls to pause, resume and stop execution. Block-level and directive-level controls exist to pause, resume and skip execution. Icons are used to show the user whether a block is paused or skipped.

SM anomaly messages that require a user response are displayed in a separate window. A synopsis is displayed in a scrolling portion at the top of the window. By selecting a synopsis, the operator brings up a description of the anomaly in the bottom portion of the window. The operator can then enter the requested input or select a default option.

Other available displays are a scrolling event log, the link configuration display and the device state display. The scrolling event log lists all the input and output to and from the LMCOA system. The link configuration display lists the equipment being used for the pass. (A link is the set of equipment assigned to communicate with the spacecraft. There may be duplicate pieces of the same kind of equipment.) The device state display shows the expected and actual state of each piece of equipment. It provides the operator with a view of the SM device model.

Knowledge Engineering

A major effort in the LMCOA is the process of gathering the information for the TDN. When we started there was no definitive answer as to what is the correct sequence of operations for doing VLBI on a 70-meter antenna, Rather, each operator had his/her own "cheat" sheet of what needed to be done. There were also official documentation, engineers' procedures for VLBI and scientists' procedures for VLBI; all of which differed from group to group and from person to person. The approach for knowledge engineering was to first learn about the system through existing documentation, noting inconsistencies and missing information. The next step was to discuss the procedure with the operators, engineers, technicians and scientists to get their viewpoints and to clear up inconsistencies as much as possible. This information led to our initial TDN. The TDN became the needed common language between our knowledge engineers and our knowledge sources. The LMCOA knowledge engineering effort is the only known attempt, within the DSN, to document a single, coherent and consistent base-line operational sequence for precalibration that merges the viewpoints of the users.

Current Implementation Status

The LMCOA is implemented using C, Objective-C and the C Language Integrated Production System (CLIPS) on a NeXT workstation running Mach. Most of the features described above have been implemented and successfully tested and demonstrated in a laboratory environment with the DSN Subsystem Simulator. Items currently being implemented are the integration of the recovery subnet, the Timed-Events Manager, automated parameterization and report generation, the link configuration display and the device state display. Our plan is to demonstrate the LMCOA prototype at the Goldstone Deep Space Communications Complex's 70-meter antenna in September, 1992.

Conclusion

Knowledge-based systems will play a major and enabling role in improving future ground information systems at the DSN. The LMCOA prototype demonstrates the feasibility and benefits of AI-based automation in DSN operations. The benefits of an operational semi-automated monitor and control system are 1) reduction in precalibration time by at least 50% which could provide an extra 24 hours of antenna availability per week; 2) reduction in keyboard entry by 80% which reduces occurrences of typographic errors; 3) enablement of parallel operations ; and 4) increased operator efficiency via closed loop control. Future efforts will include extending the LMCOA to control multiple activities, demonstrating the LMCOA at a 34-meter experimental antenna complex, extending the LMCOA to support other operations and establishing guidelines for subsystem design to facilitate automated monitor and control.

Acknowledgments

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The other members of the LMCOA team are Sanguan Chow, Lynne Cooper, Kristina Fayyad and Juan Urista. We would like to acknowledge the contributions of Pam Wolken, Terry Dow, Dave Girdner, Elmain Martinez and the many operations personnel at Goldstone Deep Space Communications Complex.

References

1. Cooper, Lynne P, Desai, Rajiv and Martinez, Elmain, "Operator Assistant to Support Deep Space Network Link Monitor and Control," SOAR Symposium, Houston, TX, 1991.

2. Deep Space Network Information System Architecture Study", JPL Publication D-8916, Jet Propulsion Laboratory Internal Document, September 27, 1991.

3. Miller, Gary and Walrath, Kathy, "NeXT Operating System Software", NeXT Computer Inc., Redwood City, California, 1990.

4. Cooper, Lynne P, "Operations Automation Using Temporal Dependency Networks," Technology 2001, San Jose, CA, December 3-5, 1991.

BOOSTER EXPERT SYSTEM

Tom Kalvelage
NASA/Johnson Space Center
2101 Nasa Road 1
Houston, TX 77058

Abstract unavailable at time of publication.

DESSY: Making a Real-Time Expert System Robust and Useful

Sherry A. Land*
Jane T. Malin
NASA Johnson Space Center
Intelligent Systems Branch - ER22
Houston, TX 77058
(713) 483-2064

Donald R. Culp
Rockwell International
Remote Manipulator Section - DF44
Houston, TX 77058
(713) 483-0891

ABSTRACT

As the complexity and expected life-span of modern space systems continue to increase, the need for real-time data monitoring and failure analysis becomes more critical to their successful operation. DESSY, or DEcision Support SYstem, is a joint effort by the Intelligent Systems Branch/ER2 and the Remote Manipulator System (RMS) Section/DF44 to develop an expert system for the monitoring of the Payload Deployment and Retrieval System (PDRS). DESSY users, the RMS flight controllers, are provided with user interface enhancements and automated monitoring of system state (physical orientation) and status (operational health). Currently, a DESSY prototype for the Manipulator Positioning Mechanism (MPM) and Manipulator Retention Latches (MRL) of the PDRS has been developed and successfully demonstrated during the STS-49 and STS-46 missions.

Expert systems for monitoring real-time operations must not only accurately represent domain knowledge, but also address the challenges of using unfiltered real-time data as input. This paper describes the methods and design strategies developed to overcome problems with real-time data in the NASA Mission Control Center. Types of data problems addressed are (1) loss of data, (2) erratic data and (3) data lags and irregularities during state transition. Methods used to handle data problems include rule disabling for ignoring data when data quality is uncertain, context-sensitive bounded pattern recognition for minimizing incorrect conclusions based on bad data, and graceful recovery through system correction when reliable data returns. This combination of methods with an object-based modular DESSY design assures a robust program capable of lengthy periods of uninterrupted use in operations.

I. INTRODUCTION

As modern space systems become more advanced, the need for intelligent computer-assisted support during operations continues to grow. The support required usually begins with real-time data monitoring and includes failure diagnosis and possibly recommended actions. As a result intelligent software must continue to evolve to meet these demands. In this paper we discuss the development of the DEcision Support SYstem (DESSY), an expert system which aids flight controllers by performing real-time data monitoring and intelligent evaluations of system hardware through assessments of telemetry data patterns and transitions. Although we are documenting many aspects of the development process, we focus this discussion on the most challenging issue we faced - dealing with unreliable real-time telemetry data. We briefly address other development topics including knowledge base organization, object and rule structure and user interface enhancements.

II. THE DESSY EXPERT SYSTEM

Project Background

The DESSY project is a joint effort by the Intelligent Systems Branch/ER2 and the Remote Manipulator System (RMS) Section/DF44 of Johnson Space Center to develop an expert system for monitoring the Payload Deployment and Retrieval System (PDRS). The PDRS is made up of several subsystems, each of which will correspond to a DESSY module. Currently, a DESSY prototype for the Manipulator Positioning Mechanism (MPM) and Manipulator Retention Latches (MRL) of the PDRS has been developed and successfully demonstrated during the STS-49 and STS-46 missions. MPM's are pedestals that roll the shuttle arm in and out of the payload bay. MRL's are latches which latch the arm in place when it is not being used. The next subsystem being undertaken is the Remotely Operated Electrical Umbilical (ROEU), a system to provide an electrical interface between the orbiter and a payload while the payload is in the payload bay. Other RMS subsystems to be implemented as DESSY modules include the End Effector, Arm-Based Electronics, and Displays & Controls Panel.

C-C

Software Design

DESSY is implemented in a commercial real-time expert system development environment. Its object-oriented data structures represent the RMS system design and its rules capture the logic of system operations and failures. Modularity within the software separates rules for data monitoring, state transition detection, failure diagnosis, expert system control and user interface. This partitioning of rules allows the enabling or disabling of appropriate groups of rules when necessary, as in the case when unreliable telemetry data enters the system, or for changing system contexts. For example, as the MPM/MRL system reaches a new configuration in nominal operations, state transition rules monitor this change of state and DESSY changes to a new context. As a result, the appropriate set of diagnostic rules is enabled for the duration of this context.

The DESSY software design is further modularized in that each RMS subsystem will be implemented as a separate software module, able to act independently of the others. Each subsystem module will have a rule set and display developed specifically for that RMS subsystem. A summary display called the Integrated Status Display will provide an overview of all subsystem states and statuses. Figure 1 shows how each of the planned subsystem modules fits in the overall DESSY design.

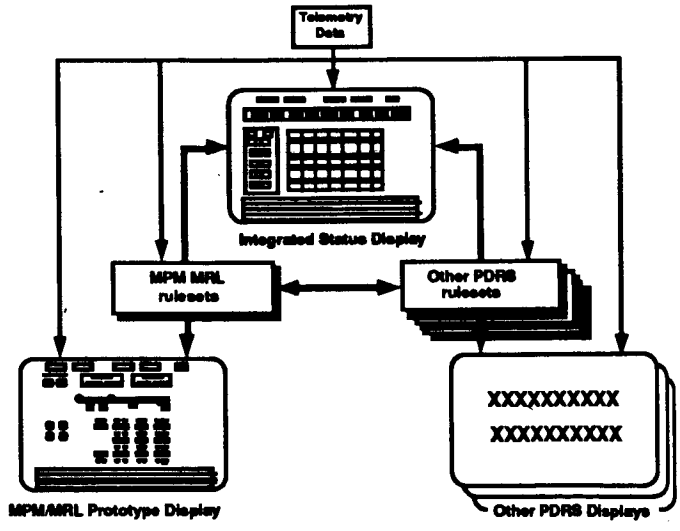


Figure 1. The DESSY Design

Careful object and rule design, along with appropriate modularity, both in subsystem partitioning and in the separation of rules according to rule functionality, has played an important role in the successful development of DESSY. However, even if the software provides objects and rules that correctly capture the functionality and logic of the physical system, failure to address the issues concerning real-time data as input can lead to disappointing system behavior. In the remaining section of this paper we discuss problems observed and solutions developed for working with real-time data. It is these methods that have contributed most to the robustness of DESSY.

III. WORKING WITH REAL-TIME DATA

Although many expert system projects deliver a knowledge base which accurately represents the domain of the problem, they generally fail to successfully address the challenges of using unfiltered real-time data. We present several types of problems that occur when using real-time data as input to a monitoring expert system. Although all telemetry sensor data sent to DESSY is binary, these problems could occur with any data format. The data problems addressed include (1) loss of data, (2) erratic data and (3) data lags and irregularities during state transitions. Each of these problem types, discovered during DESSY development and testing, is discussed.

Following the overview of data problems, we examine the solutions developed to account for these problems. Table I lists each of the data problems with their applicable solutions. These methods, in the order of increasing sophistication and potential benefit to real-time operations, include rule disabling to ignore data when data quality is uncertain, context-sensitive bounded pattern recognition for minimizing incorrect conclusions based on bad data that has entered the system, and graceful recovery through system correction when reliable data returns. Each method works independently to prevent or correct erroneous expert system conclusions resulting from bad or missing data. It is their combination, however, that assures a robust program capable of lengthy periods of uninterrupted use in operations.

Data Problem	Solution Methods
loss of data	rule disabling, graceful recovery
erratic data	pattern recognition, graceful recovery
data lags and irregularities during state transition	pattern recognition, graceful recovery

Table I. Data Problems and Solutions

Types of Data Problems

Loss of Data

The most common and well understood data problem is a loss of data. Data loss usually takes the form of a complete loss of signal (LOS) and may be either expected, i.e., the shuttle enters the Zone of Exclusion (ZOE) where there is no telemetry downlink, or it may be unexpected. Unexpected data loss may occur as a complete or partial loss of data, usually due to ground processing or computer hardware problems.

Although LOS seems like a simple concept to account for, hardware implementation of telemetry processors can complicate the situation. Depending upon how a particular telemetry processor handles periods of LOS, the monitoring system may receive an inactive state for all data values, no data values at all, a static frame of the last data values, or as in the case of the DESSY data source, the Real Time Data Systems (RTDS), the last 4-5 seconds of data may be repeatedly replayed until the signal returns. Yet another possibility is that nonsense data is received during periods of LOS. The first step in dealing with loss of data due to signal loss is to find out the form of data that will be received during this time.

Once it is understand what LOS will mean for a monitoring expert system, it must be determined how that system is going to detect it. One possibility is to compare actual data format with expected format for each data frame received to determine quality. In the DESSY project we were fortunate enough to have a data quality measurement from the telemetry processor. OI-Quality indicates the telemetry processor's assessment of data quality for each data frame. It was discovered, however, that OI-Quality did not always reliably reflect true data quality. Often there seemed to be a lag between the data quality drop and OI-Quality's reflection of this drop. At other times, even when the lag did not appear, the low quality indication occurred in the same data frame that included bad or missing data, making it impossible to filter the data or to alert the system of its presence. Inevitably, bad data due to an LOS situation would periodically enter the system.

Later in the paper we present the methods developed for dealing with LOS problems, such as disabling of rules and recovery methods. These methods apply to both the case when LOS is detected in time to prevent problems from occurring and when bad data due to LOS enters the system.

Erratic Data

Erratic data is unstable and does not meet expected behavior for a given operational context. Erratic data may occur at anytime, but is most likely to be seen immediately before or after an LOS or at the time of state transitions. Erratic data is characterized by frequent flipping of bits (in the binary case) in a particular data set. Bit flipping occurs when a data value toggles or flips between values of 0 and 1. Of course this signature may also result from intermittent sensor failures, and that possibility should not be ruled out. For large sets of data, however, it is much more likely that any bit flipping is due to bad telemetry data, rather than bad sensors.

For DESSY, periods surrounding an expected LOS seem to be a common time for erratic data to appear. As the shuttle moves in or out of a satellite's range, the telemetry link has a period of degradation, during which the OI-Quality has not yet dropped. The result is often a significant amount of bad data entering DESSY. Because there has been no previous low quality indication, DESSY has no clue that a data frame from this scenario contains degraded data.

The second situation when erratic data often occurs is near the beginning and end of a state transition, when DESSY frequently encounters bit flippings of data. Although this is an instance of erratic data, this scenario is discussed in the following paragraphs concerning data behavior during state transitions. Later we discuss our use of context-sensitive bounded pattern recognition and graceful recovery to deal with both these types of problems.

Data Lags & Irregularities During Operational Events

The final type of data problem we discuss results from unexpected data activity when data is expected to change because of an operational event such as a state transition or commanding. Data that is expected to change may "flicker" or "lag" before reaching a new stable state. Given a set of data that is expected to change at transition time T, subsets may flicker or lag, causing the data transition to occur over some delta time t. Typically delta t is 1-3 seconds. The following transition graph depicts five examples of data transitions from low to high values, including a normal data transition and four anomalous cases. The delta time for this data set is 2 seconds because it is the time it took for every piece of data in the set to change to its new expected state.

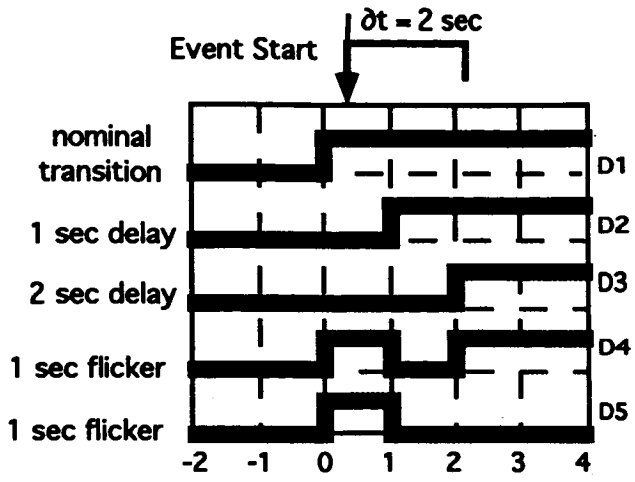


Figure 2. Examples of Data Lags and Irregularities

Data lags and irregularities during events and state transitions may occur because of noise in the telemetry or may be due to the physical properties of the hardware being monitored. In either case there is no smooth transition. Failing to include this behavior in the system's monitoring rules will often lead to erroneous conclusions. Examples from early DESSY work include (1) incorrect sensor failure conclusions at Event Start T due to data lags - D2 and D3, (2) state value fluctuation from T to T+2 from flickering data - D4, and (3) incorrect command conclusion at time T due to a temporary active value in command telemetry - D5. These problems were corrected using our data handling techniques.

Methods Used to Handle Data Problems

Rule Disabling

The most straightforward method of dealing with data of uncertain quality is to ignore it by not responding to changes in that data. In any system there will be times when data quality is so low that the data should not be used at all. The expert system should therefore have the capability to ignore data when it is unreliable by a method such as disabling of rule sets. This immediately highlights the need for a manner of determining faulty data, such as the OI-Quality measure provided to DESSY. When the value of OI-Quality is any number other than 100, certain diagnostic and state transition rule sets are disabled in DESSY. When quality returns the rules are again enabled.

Although the above tactic seems simple enough, it has the problem of the quality number and corresponding data being in the same time frame, making immediate filtering or rule disabling impossible. Also as the shuttle enters or leaves the Zone of Exclusion, data deteriorates, making OI-Quality itself unreliable at that time. However, even though erroneous data may have already entered the system, it is still desirable to disable rules to minimize the number of faulty conclusions. Event rules such as command and state transition, and diagnostic rules should be disabled, but corrective rules should not be disabled. Corrective rules will be discussed in the section on graceful recovery.

Disabling rules is the simplest of the techniques we use in dealing with real-time data problems. Unfortunately, it effectively eliminates the usefulness of the expert system during the time the rules are disabled, retaining only its function as a raw data monitoring source. For this reason the technique is only used when absolutely necessary - when it is certain that quality is low and data is unreliable, such as in a complete loss of signal.

To supplement this automatic rule disabling, the DESSY user may also "turn off" the expert system portion of DESSY at any time, leaving DESSY to act as only a data monitor. Actually this turning off merely grays out the parts of the DESSY display that present expert system conclusions, allowing DESSY to continue to work in the background. Even if DESSY has made incorrect conclusions and the user has grayed out the expert system part of the DESSY screen, the built in corrective rules should eventually lead to a graceful recovery. The intent is that even if DESSY has been turned off due to erroneous expert system conclusions, it will recover by itself and the user will once again be able to use the expert system part of DESSY. Nonetheless, this feature gives the user the opportunity to override the expert system at the level of the user interface.

Context-Sensitive Bounded Pattern Recognition

The second technique we have implemented to assure DESSY's robustness deals with characteristics of the sets of data that DESSY's rules use to detect events and identify failures. Because of problems with data lags and irregularities, the expert system often has insufficient or even erroneous evidence from the data set upon which it can determine the occurrence of an event. Also because of the nature of space operations, there is often an insufficient amount of sensor data available, making event determination even more difficult. For example, if the event is that of an *MPM stow*, there are only two sensors to indicate the stowed state once the MPM has reached its new stowed position. This data set is insufficient to determine with certainty the stowed state because of a requirement we have that DESSY must continue to monitor events, given a single data failure in any set of data

we consider. If one of these two pieces of data was active and the other inactive, as is the case of a single failure of either of these pieces of data, the state would be inconclusive. The data must therefore be supplemented with additional information.

In addition to considering data set size, when appropriate we include context such as the system state or the detection of a prior event. When context is considered, we say that we are using a context variable (CV) in the rule. Use of CV's lowers the requirements of the data set the rule must consider, and some rules use only CV's. Table II gives an example of a DESSY rule using two pieces of data and two context variables.

We now discuss the general guidelines developed in determining necessary data sets and context variables for DESSY rules. The lower bound or minimum requirements needed to make reliable conclusions is addressed first, followed by a brief discussion on the upper bound or maximum data set recommended. Establishment of a lower bound is necessary because enough data must be observed to correctly monitor an event, given that some predetermined amount or percentage of the data set being considered is bad. An upper bound is established because of the impact the data set size has on computer hardware performance. An example from DESSY is provided.

DESSY Rule: state transition*	Data/CV set
if the state of any MPM is *stow-in-transit*	CV
and (the sys1-stow-microswitch = *1*	data
or the sys2-stow-microswitch =*1*)	data
and the command of MPM-SYSTEM is *stow*	CV
then conclude that the state of the MPM is *stowed*	conclusion

Table II. Rule Example with 2 Pieces of Data and 2 Context Variables

Establishing a Lower Bound

Given an operational event there exists a set of data S that the expert system directly monitors to determine when the event occurs. In addition, there exists a second usually larger set S', a superset of S, that makes up the context in which the event will occur. The set S' indicates state, status and any other operational context of the system being monitored and includes both data and context variables.

The CV's from S' in the example of Table II are the current MPM state and the current MPM command. Thus we have the following sets.

S_{stowed} =
{sys1-stow-microswitch, sys2-stow-microswitch}

S'_{stowed} =
{sys1-stow-microswitch, sys2-stow-microswitch,
current state, command}

If the set S were the only data we could consider, we would have to impose the constraint that both data elements be active to eliminate the ambiguity and ensure that we were in a stowed state; however, this does not meet DESSY's requirement to continue to monitor given a single failure. Because in the extended set S' context is considered, we can relax our constraints to require only one of the two microswitches to be active in determining the stowed state. We have $|S_{stowed}| = 2$ and $|S'_{stowed}| = 4$ where $|S|$ denotes the number of elements in the set S. Because the set size of 2 is insufficient for determining the event given our requirements, the data set S must be expanded to S' providing a more plausible set.

Even in the case where 3 pieces of telemetry are available, the data is probably insufficient. Although in actual operations, a double hardware failure must occur to lose 2 of the 3 sensors, in a situation where noisy data is occurring, the possibility that 2 of these 3 may erroneously become "turned on" is fairly high. In this situation a larger data set including context variables is desirable. CV's such as state and command in the example above, can be used to obtain the minimum information set by imposing physical constraints or providing the current system configuration. CV's limit the scope of a rule which in turn limits the chance that it will fire incorrectly given it has received bad data.

Although we use CV's to impose physical constraints in DESSY, procedural constraints are not used in DESSY rules, since humans are too likely to break procedural rules. We learned this while testing an earlier DESSY version when a flight rule was broken during a training simulation. The software failed to monitor the operations due to a procedural constraint embedded in DESSY's monitoring rules. Therefore, as a policy, we implement no procedural constraints in DESSY.

Establishing an Upper Bound

In real-time operations, every piece of data observed has an associated cost in CPU time. Thus we wish

* MPM Stowing is the rolling in of the RMS to put it away. Deploying is rolling out the RMS.

to impose limits on the amount of data DESSY is allowed to inspect in a given rule. The best way to implement this is through context variables which hold summaries of data values and are usually already stored in DESSY for other purposes. Use of CV's allows us to consider performance constraints to limit the size of the data sets we inspect. In some cases a CV can take the place of all but one piece of data. In other higher level rule firings, such as for summary level information, conclusions are made based only on context variables.

We have established the upper bound of a rule as 4 pieces of telemetry data or 2-3 pieces of data given a CV. The real-time expert system should *usually* not be overloaded with a larger set, although there will probably be exceptions in safety critical or unreliable areas. In conclusion, context-sensitive pattern recognition with appropriate set size, including both data and context variables, reduces the chances of incorrect expert system behavior when bad data is present.

Graceful Recovery Through System Correction

Although many systems attempt graceful degradation in the face of trouble, DESSY has extended this concept to one of graceful recovery. If the system makes faulty conclusions because of bad data, a set of corrective rules will "kick in" once good data returns, and restore the expert system to the proper configuration. This includes both state correction and status correction. The system does not have to be restarted by the user because the corrections automatically restore offending parts of the knowledge base.

Corrective rules are similar to other system monitoring rules, except that they do not allow for any inconsistencies in the data sets they observe when making conclusions, i.e., every piece of data in the set for which the rule applies must be exactly correct. Additionally, corrective rules are written only for the cases that it can be determined with certainty that the system is in a particular configuration. These rules, therefore, can be thought of as the axioms of the expert system. They can be as simple as determining that a single piece of data is reliable again because it returned to a legitimate value after it had previously been deemed as unreliable. A more sophisticated example is the re-evaluation of system state when all microswitch data for a new state becomes active.

Examples of corrective rules are provided in Figures 3 and 4. The rule in Figure 3 shows the re-evaluation of the status of a single microswitch. If the microswitch status is *questionable-on* because the microswitch is inappropriately active, but the microswitch value returns to inactive (0 or off), the status is reset to *functional*. Figure 4 is an example of the re-evaluation of the state of MPM's. If the state is not stowed, but both stow microswitch indicators become active (1 or on), then the MPM state is reset to stowed.

If the status of sys1-stow-microswitch is questionable-on
and the sys1-stow-microswitch = 0
then conclude that the status of sys1-stow-microswitch is functional

Figure 3. Corrective Rule: Microswitch Status

if the state of any MPM is not stowed
and the sys1-stow-microswitch = 1
and the sys2-stow-microswitch = 1
then conclude that the state of the MPM is stowed

Figure 4. Corrective Rule: MPM State

The example of Figure 3 may occur when an MPM stow microswitch (normally indicative of the stowed state) becomes active while the MPM is in the deployed state. This event could be due to a microswitch failure, but is more likely to be caused by bad telemetry. When this event occurs, several conclusions are made. First the microswitch is determined to be *questionable-on*. Assigning this value is a strategy to allow the user to realize DESSY is at this point making only a tentative conclusion about a failure of this microswitch. If the microswitch remains active, it will cause a motor to be inhibited, preventing that motor from driving if a stow command is given. (However, there are two redundant motors for stowing, and the other motor will still operate.) Secondly, because the MPM is deployed and a stow microswitch is *questionable-on*, the status of the MPM will be updated to *Expect-Single-Motor-Stow*. If before stow operations occur, the microswitch returns to its inactive state, *questionable-on* is removed by the rule in Figure 3 and the MPM status is restored to operational.

If stowing operations proceed and the motor is inhibited by the microswitch, procedure monitoring rules will notice this fact. The MPM status will be updated to *Single-Motor-Stow* once the stow is complete and it has been confirmed that only one motor was operational. However, if the motor is not inhibited, but the *questionable-on* microswitch was due to bad telemetry, procedural monitoring rules note that the motor performed nominally and the MPM status is adjusted to *Nominal-Stow*. The microswitch status, however, will remain as *questionable-on* until the microswitch value changes to inactive.

A benefit of self correction is demonstrated in a scenario observed during STS-49. Because of hardware problems, a significant amount of data frames were being lost, so that DESSY did not receive data for seconds at a time. Unfortunately, those were crucial seconds in which an MRL state transition was taking place. DESSY was unable to monitor the transition procedure because it never received the data. Fortunately when DESSY did again receive a data frame, although the procedure was over, it was able to evaluate the current data and reflect the new MRL state. Another example of why graceful recovery is crucial for real-time expert systems occurred during MRL release operations during STS-46. Seconds after the release began, an LOS occurred and the telemetry downlink was lost. This LOS lasted for approximately seven minutes, and when data returned the release was complete. At this time DESSY evaluated the new data and reconfigured itself to reflect the new MRL state. In both these cases, it was unfortunate that we were not able to monitor the operations, but the fact that DESSY was able to keep up once data returned prevented a situation where the wrong state would be reflected, certainly causing a loss of user confidence.

Additional examples of how the corrective rules were crucial to system success were demonstrated near other times of LOS. Because poor data quality is not always detected soon enough, unfiltered erroneous data enters the system and leads to faulty conclusions. Even if it were possible to filter out the bad data as soon as it was detected, the use of unfiltered data is desirable because it gives the flight controller a better understanding of the downlinked telemetry. The expert system interpreting the data, however, is unlikely to be correct and should provide an assessment that its interpretation is suspect. Regardless, once quality is nominal again, the expert system should re-evaluate the scenario and adjust any faulty conclusions it made. State should be re-evaluated and any status discrepancies that are no longer accurate should be removed.

There is a final area in which corrective rules are utilized. This case takes place when a real failure (or even a single data or sensor failure) occurs and is later followed by recovery. The corrective rules are used to restore the status to nominal in these situations. In these cases when a failure actually has occurred, the human would like to know the failure history even if the failure is corrected. (This is currently a future DESSY version enhancement.) A history is kept in the form of a message list at present.

We have found these correction features not only to be very useful, but to have good side effects. Complementing the correction rules (and often actually the same rule), initialization rules allow DESSY to be started with any stable MPM/MRL configuration and to be initialized to the proper states and statuses. The rule in Figure 4 would initialize DESSY's MPM state to *stowed* if DESSY were started with the MPM in the stowed position. If DESSY is started during transition periods, once the transition is complete and the system is again static, DESSY will at that point initialize itself. This has been an important and even necessary feature of our real-time expert system.

In summary, it is necessary to keep the system from making lasting erroneous conclusions based on bad data. When working with real-time data, we must expect the unexpected in data problems. An attitude must be maintained that in spite of filtering and other techniques to keep bad data out of the system, some will always get through. When it does the only choice is to design for recovery when things return to normal.

IV. CONCLUSIONS IN MAKING DESSY ROBUST & USEFUL

In building real-time expert systems, it is not only important to have good system design, but also crucial to create a system that is robust enough to be useful in situations that are less than ideal. The techniques developed through the DESSY project were created out of the necessity of building a system that could withstand bad data, a common occurrence in the environment in which DESSY runs. These methods have been successfully used in DESSY's operation and have repeatedly demonstrated DESSY's robustness.

In conclusion, real-time data monitoring systems that reside in high-risk environments such as NASA's Mission Control Center must be built to be robust and useful given bad and missing data. Although the specifics of these techniques will differ from one project to another, we believe they are general enough to be applied to any real-time monitoring expert system, and that the guidelines presented in this paper provide a good set of ground rules from which a robust and useful system can be built.

BIBLIOGRAPHY

Collins, David, "Payload Deployment and Retrieval System Overview Workbook," PDRS OV 2102, Mission Operations Directorate, Johnson Space Center, Houston, Texas, February, 1988.

Dvorak, Daniel, "Expert Systems for Monitoring and Control," AI 87-55, The University of Texas at Austin, Austin, Texas, May, 1987.

Malin, Jane, Schreckenghost, Debra, and Thronesbery, Carroll, "Design for Interaction Between Humans and Intelligent Systems During Real-Time Fault Management," Proceedings of Fifth Annual Space Operations, Applications, and Research Symposium, NASA Johnson Space Center, Houston, Texas, July 9-11, 1991.

Malin, Jane, Schreckenghost, Debra, et.al., "Making Intelligent Systems Team Players: Case Studies and Design Issues," NASA Technical Memorandum 104738, NASA Johnson Space Center, Houston, Texas, September 1991, pp. 530 - 538.

Moore, Robert, and Kramer, Mark, "Expert Systems in On-Line Process Control," Lisp Machines, Inc., Los Angeles, California.

PERTS: A Prototyping Environment for Real-Time Systems†

Jane W. S. Liu, Kwei-Jay Lin and C. L. Liu
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

ABSTRACT

PERTS is a prototyping environment for real-time systems. It is being built incrementally and will contain basic building blocks of operating systems for time-critical applications, tools and performance models for the analysis, evaluation and measurement of real-time systems, and a simulation/emulation environment. It is designed to support the use and evaluation of new design approaches, experimentations with alternative system building blocks, and the analysis and performance profiling of prototype real-time systems.

I. INTRODUCTION

While existing approaches, techniques and tools for the design, prototyping and development of software systems are effective for many application domains, they often do not address the difficulties in building hard real-time computing systems. A *hard real-time computing system*, hereafter simply called a real-time system, is one in which most tasks have hard timing constraints. Here, the term *task* refers to a basic unit of work. A task may be a granule of computation, a unit of data transmission, a file access, or an I/O operation, etc. The simplest timing constraint imposed on a task is its *deadline*, the point in time by which the task is required to complete. The result produced by a task with a deadline is correct only if it is available by the deadline, in addition to being functionally correct. A late result is of little or no use. Applications supported by real-time systems include command and control, guidance and navigation, flight control, object identification, autonomous vehicle control, and intelligent manufacturing.

The approach that has been taken traditionally to construct real-time systems is to develop the application software first and then tune the application and the underlying system to make sure that all the timing constraints are met. This approach tends to produce brittle, difficult-to-modify and hard-to-maintain systems. Small changes in the application software, or in the underlying hardware and software support, can produce unpredictable timing effects that can be detected and corrected only through exhaustive testing and performance tuning. Consequently, it is costly to develop and validate new systems and to enhance, extend or port existing systems.

The lack of effective methods and tools for building robust and provably responsive real-time systems has motivated the recent research on the theoretical foundations of real-time computing [1,2]. A goal of this research is to find methods for predicting the timing behavior of the basic building blocks and the overall real-time systems built from them. Tools that support systematic construction and evaluation of real-time systems can be built based on these methods. Another goal is to develop integrated approaches to building real-time systems, as alternatives to the traditional approach. An integrated approach would begin with models and optimality criteria that explicitly account for the constraints and possibilities of trading off between various figures of merit, and then design the application and the underlying system to achieve the desired tradeoffs. Such an approach would lead to more flexible, easy-

† This work has been partially supported by ONR Contract Nos. NOOO14-89-J-1181, and N000-92-J-1815.

to-schedule programs, and resultant systems would degrade gracefully during overloads and failures.

This paper describes an ongoing project to build a prototyping environment, called PERTS (Prototyping Environment for Real-Time Systems), that aims at making recent and future theoretical results in real-time systems readily usable to practitioners. Specifically, PERTS will contain

(1) basic building blocks of the underlying support system for real-time applications — These reusable building blocks will realize existing and new scheduling algorithms, communication protocols and resource access control protocols.

(2) building blocks of flexible real-time programs and system software — These system components are based on the imprecise computation approach [3-5].

(3) tools and performance models for the analysis and evaluation of prototype real-time systems — The PERTS tools will provide worst-case bounds and performance predictions of systems based on different execution models and scheduling paradigms.

(4) a simulation/emulation environment — This environment will allow the experimental evaluation of alternatives in scheduling the target software system.

The rest of the paper is organized as follows. Section II describes the models of real-time systems on which PERTS components and tools are based. The capabilities of PERTS and its key components are presented in Section III. This project is compared with similar projects in Section IV. Section V is gives the current status of the project.

II. MODELS OF REAL-TIME SYSTEMS

Most of the workload models used to characterize real-time (software) systems are variations or extensions of the following basic deterministic model. The underlying system contains a number of identical processors. The software system **T**, called a *task system*, contains a number n of tasks. The maximum amount of processor time required by a task T_i to complete its execution is called its *processing time* τ_i. τ_i is assumed to be known. Tasks may have weights which tell us how important the tasks are relative to each other. Again, a task T_i may have a deadline d_i; we say that a task has no deadline if its deadline is infinite. In addition to its deadline, a task T_i may also have a release time r_i, the time instant after which the task is available to be scheduled and executed. The interval $[r_i, d_i]$ between its release time and deadline is its *feasible interval.*

A *schedule* of a task system **T** is an assignment of the processors to the tasks in **T**; a task is scheduled in a time interval on a processor if the processor is assigned to the task in the interval. In any valid schedule, every task is scheduled after its release time. Moreover, the total amount of processor time assigned to every task is equal to its processing time. A valid schedule is a *feasible schedule* if every task is scheduled in its feasible interval and, hence, completes by its deadline.

The system may also contain a number of distinct resources. Each task may require some of these resources during its execution. We say that tasks requiring the same resource are in (*resource*) *conflict* with each other. A resource access control protocol governs the accesses of tasks to resources and resolves the conflicts among them.

Periodic-Task Model

Many real-time applications, such as control-law computations, radar signal processing, and voice/video transmissions, can be characterized by the classical *periodic-task model* [6]. In the periodic-task model, we model such computations and data transmissions as *period tasks*. The system **T** contains

n periodic tasks, each of which is a periodic sequence of requests for the same work. A request is released at the beginning of every period and its deadline is the end of the period. The processing time τ_i of T_i is the maximum amount of processor time required to complete every request in T_i.

In addition to periodic tasks, some tasks may arrive and become ready for execution at random instants. These tasks are *aperiodic*. Aperiodic tasks model computations and communications that must be carried out in response to unexpected events, such as requests for changing the operation mode, processing sporadic messages, etc. Aperiodic tasks usually do not have deadlines, and their processing times may be unknown. We want to complete each aperiodic task as soon as possible, while making sure that all deadlines of periodic tasks are met at all times.

Complex-Task Model

Real-time tasks that are not periodic are often characterized by the classical deterministic model. In this model, a task system **T** is a set of n tasks. These tasks may be dependent; data and control dependencies between tasks impose constraints on the order in which tasks are executed. We use a *precedence relation* $<$ over **T** to specify the constraints on their execution order. T_i is a *predecessor* of T_j (and T_j a *successor* of T_i), denoted as $T_i < T_j$, if T_j cannot begin execution until the execution of T_i completes. T_i is an *immediate predecessor* of T_j (and T_j is an *immediate successor of* T_i) if $T_i < T_j$ and there is no task T_k such that $T_i < T_k < T_j$. Two tasks T_i and T_j are *independent* when neither $T_i < T_j$ nor $T_j < T_i$. They can be executed in any order. We can use a directed graph $G = (\mathbf{T}, <)$, a *task graph*, to represent the task system **T** and the precedence constraints among tasks. There is a node in G for each task in **T**. There is an edge from T_i to T_j when T_i is an immediate predecessor of T_j. Figure 1 shows a task graph for example. Nodes of all shapes represent tasks. The numbers in the brackets above the tasks are the feasible intervals of the tasks. For simplicity, their other attributes, such as their processing times and resource requirements, are not shown.

We note that a periodic task in the periodic-task model can be modeled as an infinite chain of dependent tasks where the first task is the immediate predecessor of the second task, the second task is the immediate predecessor of the third task, and so on. Such a chain is shown in Figure 1; it represents a periodic task whose first request is released at time 2 and whose period is 3.

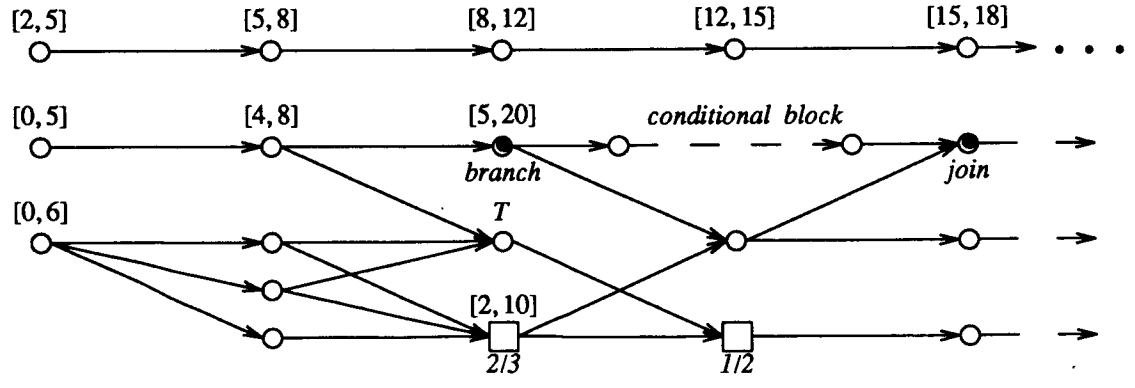


Figure 1 An Example of Task Graphs

Real-time applications sometimes contain redundant modules, carry out heuristic searches, use multiple versions, execute some tasks conditionally, etc. These applications cannot be conveniently characterized by the classical model. For this reason, we extended the classical model; the extensions include OR tasks [7] and conditional blocks [8].

In the classical model, a task with more than one immediate processor must wait until all its immediate processors have been completed before its execution can begin. We call such tasks *AND tasks*. An example is the task labeled T in Figure 1. All three of its immediate predecessors must be completed before T can begin execution. All other AND tasks are represented by unfilled circles. In some applications, a task may begin execution after one (or some) of its immediate predecessors is completed. Such a task is called an *OR task*. A task system containing AND and OR tasks is said to have *AND/OR precedence constraints*. Examples of OR tasks are the two square nodes at the bottom of the graph in Figure 1. The one labeled *2/3* can begin execution as soon as 2 of its 3 immediate predecessors complete. In a triple-redundant module, the voter can be modeled as a 2/3 OR task; it and its successors can proceed as soon as two out of its three replicated immediate predecessors complete. Similarly, we can model a two-version computation as the two immediate predecessors of a 1/2 OR task; only one of them needs to be completed before the OR task can begin execution.

In the classical model, all the immediate successors of a task must be executed; an outgoing edge from every node expresses an *AND constraint*. This model cannot characterize data-dependent, conditionally executed tasks. In the complex-task model, some outgoing edges express *OR constraints*. Only one of all the immediate successors of a task whose outgoing edges express OR constraints is to be executed. Such a task is called a *branch* node. In a meaningful task graph, there is a *join* node associated with each branch node. Each subgraph whose source node is an immediate successor of a branch node and whose sink node is an immediate predecessor of the corresponding join node is called a *conditional branch*. Here, by a source (or sink) node of a subgraph, we mean a node that has no predecessor (or successor) in the subgraph. The subgraph that begins from a branch node and ends at the associated join node is called a *conditional block*. Only one conditional branch in each conditional block is to be executed. An example is shown in Figure 1 where the conditional block has two conditional branches. Either the upper conditional branch, containing a chains of tasks, or the lower conditional branch, containing only one task, is to be executed.

Imprecise-Computation Model

For many real-time applications, it is better to have timely, approximate results than late exact results. A system that supports *imprecise computations* [3-5] attempts to produce usable approximate results when an overload or failure prevents an exact result from being produced in time. The system does so by trading off the quality of the results produced by the tasks for the amounts of processing times required to produce the results. To make this tradeoff possible, we structure every task in such a way that it can be logically decomposed into two parts: a mandatory part and an optional part. The mandatory part is the portion of the task that must be done in order to produce a result of an acceptable quality. This part must be completed before the deadline of the task. The optional part is the portion of the task that refines the result. The optional part, or a portion of it, can be left unfinished, if necessary, at the expense of the quality of the result produced by the task.

The imprecise-computation model captures this task structure. Each task T_i is decomposed into two tasks: the *mandatory* task M_i and the *optional* task O_i. Let m_i and o_i be the processing times of M_i and O_i, respectively. $m_i + o_i = \tau_i$. m_i is always bounded and known. On the other hand, o_i and, hence, τ_i can be unknown and unbounded. The release time and deadline of the tasks M_i and O_i are the same as that of

T_i, and O_i is the immediate successor of M_i. We note that the complex-task model is a special case of the imprecise computation model in which all tasks are entirely mandatory, that is, $o_i = 0$ for all tasks. Intelligent and incremental computations, known as anytime or sufficiently good computations in AI literature, can be also modeled as tasks that are entirely optional, that is, $m_i = 0$ for all tasks.

In a valid schedule of a system of imprecise tasks, the total amount of processor time assigned to each task is at least equal to m_i and at most equal to τ_i. A task is said to be *completed in the traditional sense* at an instant t when the total amount of processor time assigned to it becomes equal to its processing time at t. A mandatory task M_i is said to be completed when it is completed in the traditional sense. The optional task O_i may be terminated at any time, however, even if it is not completed at the time; no task is scheduled outside of its feasible interval. A task T_i is said to be completed in a schedule whenever its mandatory task is completed. When the total amount of processor time σ_i assigned to O_i in a schedule is equal to o_i, the *error* ε_i in the result produced by T_i (or simply the error of T_i) is zero. Otherwise, if σ_i is less than o_i, the error of T_i is equal to $E_i(\sigma_i)$, the *error function* of the task T_i. $E_i(\sigma_i)$ is typically a monotone non-increasing function of σ_i. In other words, the longer a task is allow to execute, the smaller the error in the result it produces.

Reference Model of Real-Time Systems

Figure 2 shows a generic model of real-time systems. The software system is represented by a task graph. As stated earlier, the task graph gives the processing time and resource requirements of tasks, the timing constraints of each task, and the dependencies between tasks. Tasks are scheduled and allocated resources according to a set of scheduling algorithms and resource access control protocols. This set of algorithms and protocols is an explicit element of the reference model as shown in this figure.

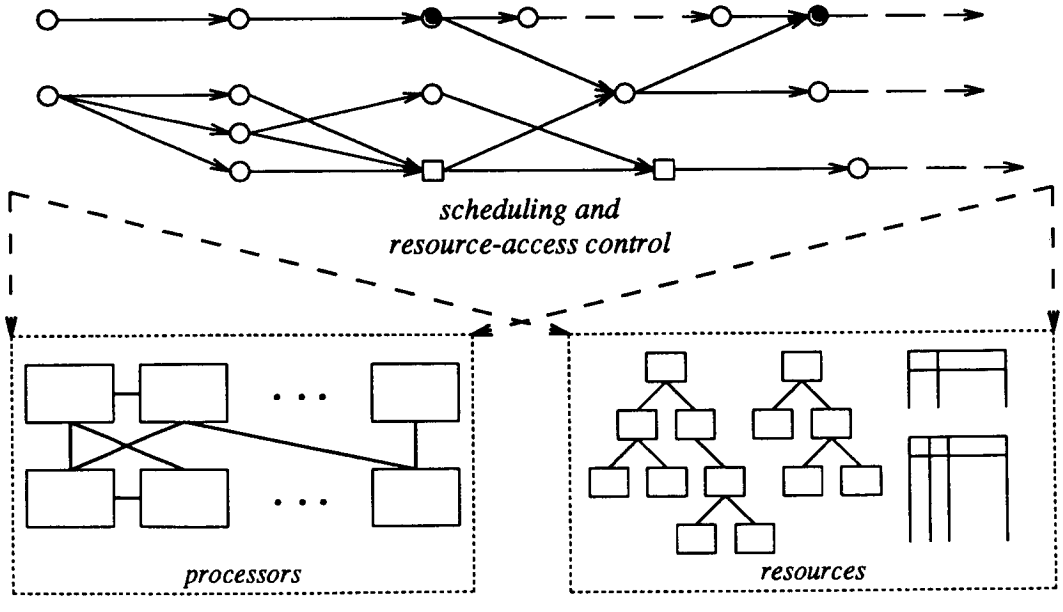


Figure 2. A Model of Real-Time Systems

The underlying hardware and run-time system is modeled as a set of processors and resources. Processors are entities that are typically modeled as servers in queuing models. Computers, I/O buses, communication networks and virtual connections are examples of processors. Resources are entities that are sometimes modeled as passive resources or passive queues. Memory pages, I/O buffers, semaphores, and valid message numbers in send and receive windows are examples of resources. It is not necessary for us to make a fine distinction between processors and resources. Rather, we characterize each processor or resource by a set of parameters. Some of these parameters specify the constraints governing its usage, such as whether it can be shared, whether it is reusable, etc. Other parameters give its timing properties, such as context switch time, acquisition time, etc.

III. PERTS COMPONENTS AND CAPABILITIES

Figure 3 shows the key components of PERTS. PERTS can be used to support the design of real-time systems. The design of a target task system is captured by its abstract description, which is a task graph. At the abstract level, estimated task parameters and dependencies in the task graph can be derived from the requirements of the system. During the design phase, the schedulability analysis system will serve as an interactive tool. This tool can be used for many purposes, including to determine whether sufficient amounts of all resources are available; to identify potential bottlenecks; to select computational algorithms from available choices with different levels of result quality versus resource requirements; and to provide suggestions on the choices of task parameters. The analysis tool and performance predictor can be used to identify where later changes in software or hardware are likely to lead to unpredictable timing effects. In this way, the schedulability analysis system can also help in the design of the test suite which will be needed later to test the target system.

The schedulability analysis system will support the hierarchical approach to building large and complex, real-time software on distributed and parallel hardware platforms. Examples of algorithms that will be implemented for this purpose include algorithms for end-to-end scheduling of distributed tasks that have overall deadlines; algorithms for scheduling parallelizable tasks with deadlines on massively parallel systems; partitioning and assignment schemes for statically assigning tasks to processors; load balancing algorithms for dynamic adjustment of load conditions; and protocols for controlling concurrent access to resources and data transmissions. For example, the task partitioning and assignment module can help the designer to find a partition and assignment of the given task system so that the tasks assigned to each processor can meet their individual deadlines and the overall task system can meet its end-to-end deadlines. When the given task system does not have such an assignment, the analysis tool in the system can suggest possible changes to make such an assignment feasible. If a dynamic task assignment approach is chosen, the performance predicting tool can be used to determine whether the worst-case performance of the assignment is acceptable.

PERTS will provide similar support in later phases of software prototyping. In earlier development stages, PERTS can be used to identify and choose a set of operating system policies for task partitioning and assignment, load balancing, scheduling and resource management. In this case, the concrete description may simply be a more detailed task graph that gives more accurate information about the timing and resource usage characteristics of the tasks. PERTS will produce sample task assignments, schedules, memory layouts, etc. to provide the feedback needed in the iterative software prototyping process. PERTS will have program execution time analysis and measurement tools. In later stages, when some source code of the target task system becomes available, these tools can be used to extract task parameters and graph structures from the code. PERTS also provides a simulation environment for a thorough evaluation of the target system. The most concrete description is the instrumented object code

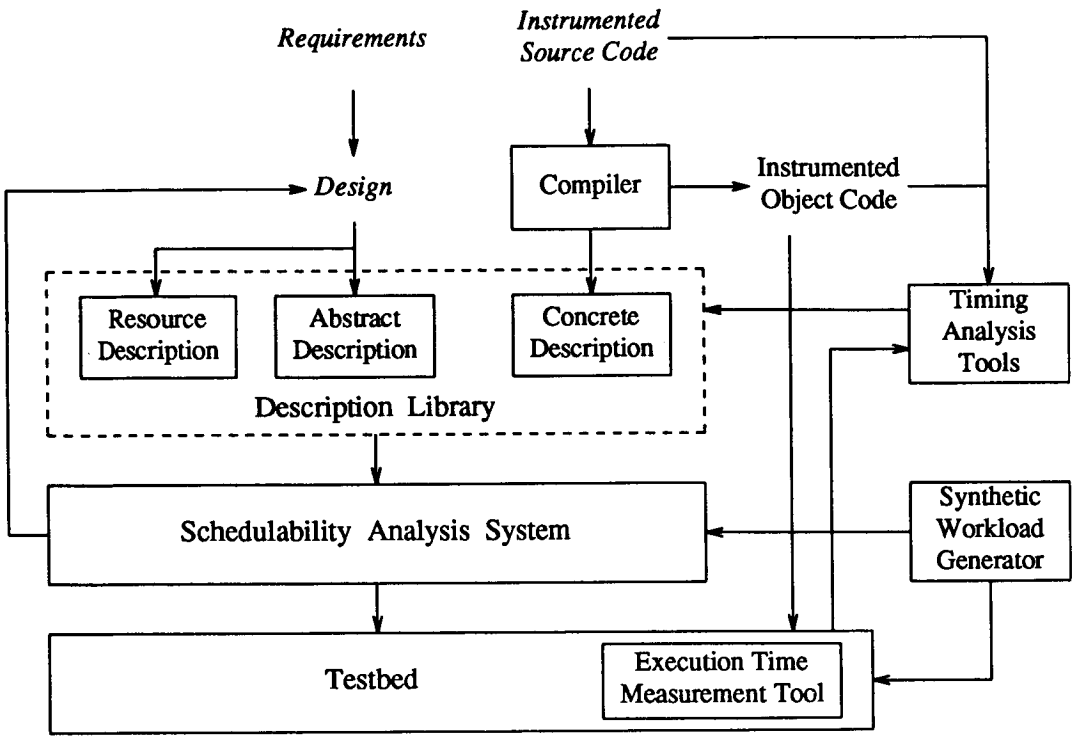


Figure 3. The Prototyping Environment of Real-Time Systems

generated by a cross language compiler. This code can be run, under the scheduling directives produced by the scheduling analysis system, in a simulated target environment provided by the testbed The testbed will contain a workload generator capable of generating synthetic or trace-driven workloads to support the simulation of the embedded environment.

IV. RELATED WORK

PERTS is similar to many other real-time systems design tools in its intended use. These systems all intend to reduce the complexity in real-time system development. The advanced algorithms and tools available in PERTS distinguish it from the other systems. For example, Scheduler 1-2-3 [9] primarily deals with periodic tasks, mixed with randomly arriving aperiodic tasks, and priority-driven scheduling disciplines. Several systems similar to Scheduler 1-2-3 are also available. They support the design and construction of domain specific applications. PERTS, on the other hand, provides a much more versatile and powerful schedulability analysis system. The PERTS testbed can be configured to simulate a number of operating systems and hardware configurations.

PERTS differs from most existing and experimental real-time system prototyping and development systems, and complements them, both in their capabilities and intended use. Many such systems provide an integrated environment with a full range of tools for requirement tracing, program construction, software reuse, etc. The experimental system CAPS [10] is an example. PERTS is similar to CAPS in certain ways; for instance, both provide tools for analysis of real-time software. CAPS is a stand-alone prototyping environment. PERTS is not designed to be a substitute for CAPS or other computer-aided

software prototyping systems. Rather, PERTS will focus on providing powerful design and evaluation tools that are not available in these systems.

V. CURRENT STATUS

We are implementing the components of PERTS incrementally in C++. Several suites of scheduling algorithms are in various stages of completion. They are algorithms for scheduling periodic tasks, imprecise computations, tasks with end-to-end deadlines [11] and tasks with AND/OR precedence constraints [7], as well as algorithms for assigning tasks to processors. The suite of algorithms for scheduling periodic tasks is near completion. Components of this suite that have been implemented and tested include the basic rate-monotone algorithm and the earliest-deadline algorithm; priority-ceiling protocol and stack-based protocol for resource access control; servers for handling aperiodic requests; and mode change protocols [1]. A basic schedulability analysis system, containing tools based on the rate-monotone scheduling theory and worst-case performance analysis, has been designed.

We have designed a simple language for describing task graphs and have implemented a compiler for this language. A user can describe a task graph in terms of this language, and the compiler will produce the graph. We also have a preprocessor that automatically extracts task graphs from annotated C++ programs.

REFERENCES

1. Van Tilborg, A. M. and G. M. Koob, *Foundations of Real-Time Computing: Scheduling and Resource Management*, Kluwer Academic Publishers, 1991.
2. Van Tilborg, A. M. and G. M. Koob, *Foundations of Real-Time Computing: Formal Methods and Specifications*, Kluwer Academic Publishers, 1991.
3. Liu, J. W. S., K. J. Lin, C.L. Liu, and C. W. Gear, "Imprecise Computations," in *Mission Critical Operating Systems*, Edited by A. K. Agrawala, C. D. Gordon and P. Hwang, IOS Press, Amsterdam, 1992, pp. 159-169.
4. Liu, J. W. S., K. J. Lin, W. K. Shih, A. C. Yu, J. Y. Chung, and W. Zhao, "Algorithms for Scheduling Imprecise Computations," *IEEE Computer*, May 1991, pp. 58-68.
5. Liu, J. W. S., K. J. Lin, and C. L. Liu, "Imprecise Computations: A Means to Provide Scheduling Flexibility and Enhance Dependability," to appear in *Readings on Real-Time Systems*, Edited by Y. Lee and M. Krishna, IEEE Press.
6. Liu, C. L. and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the Association for Computing Machinery*, Vol. 20, No. 1, January 1973, pp. 46-61.
7. Gillies, D. and J. W. S. Liu, "The Complexity of AND/OR Scheduling," *Proceedings of the 2nd IEEE Conference on Parallel and Distributed Processing*, Dallas, Texas, December 1990, pp. 394-401.
8. Kim, T., C. L. Liu and J. W. S. Liu, "A Scheduling Algorithm for Conditional Resource Sharing," *Proceedings of the IEEE International Conference on Computer-Aided-Design*, November 1991, pp. 84-87.
9. Tokuda, H. and C. W. Mercer, "A Real-Time Tool Set for the ARTS Kernel," *Proceedings of the 9th IEEE Real-Time Systems Symposium*, December 1988.
10. Luqi, "Software Evolution Through Rapid Prototyping," *IEEE Computer*, May 1989.
11. Bettati, R. and J. W. S. Liu, "End-to-End Scheduling to Meet Deadlines in Distributed Systems," *Proceedings of the 12th International Conference on Distributed Computing Systems, Yokohama, Japan, June 1992.*

PI-IN-A-BOX

Silvano Colombano
NASA/Ames Research Center
Moffett Field, CA 94035

Abstract unavailable at time of publication.

Session A7: INFORMATION MANAGEMENT

Session Chair: Dr. Jane T. Malin

The Computer Integrated Documentation Project: A Merge of Hypermedia and AI Techniques

Nathalie Mathé[1] & **Guy Boy**

NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035, USA
mathe@ptolemy.arc.nasa.gov

ONERA-EURISCO
Prologue1, BP 27-25
31312 Labege Cedex, France
boy@tls-cs.cert.fr

ABSTRACT

To generate intelligent indexing that allows context-sensitive information retrieval, a system must be able to acquire knowledge directly through interaction with users. In this paper, we present the architecture for CID (Computer Integrated Documentation), a system that enables integration of various technical documents in a hypertext framework and includes an intelligent browsing system that incorporates indexing in context. CID's knowledge-based indexing mechanism allows case-based knowledge acquisition by experimentation. It utilizes on-line user information requirements and suggestions either to reinforce current indexing in case of success or to generate new knowledge in case of failure. This allows CID's intelligent interface system to provide helpful responses, based on previous experience (user feedback). We describe CID's current capabilities and provide an overview of our plans for extending the system.

INTRODUCTION

Retrieving specific information from large amounts of documentation is not an easy task. It could be facilitated if information relevant in the current problem solving context could be supplied automatically to the user, in understandable terms and in a flexible manner (e.g., allowing the user to ask questions). This is a long-term goal of Computer Integrated Documentation (CID). As a first step towards this goal, we are developing an intelligent hypertext interface to help users browse through large documents in search of specific information [2].

CID has been developed on three aerospace applications: the Space Station Freedom Requirements documents, Space Shuttle mission control procedures manuals, and F-18 emergency procedures. These applications offer a wide variety of problems in technical documentation for both design and operations issues. This project is now three years old, and the current version of CID is used in a beta-test mode in several NASA centers. A typical screen of CID windows is presented in Figure 1. It is composed of a control panel that allows the user to control the entire library. Among its various capabilities, it also converts flat text documents into hypertext, indexes them, and records users' traces in the documentation. Both text and graphics capabilities are available within CID.

This paper provides a paradigm for information access based on the actual use of the hypertext system itself. First, it presents a new approach to incremental context acquisition in information retrieval that modifies existing relations between descriptors and referents by using on-line user feedback to either reinforce or correct the system's knowledge in case of success or failure. This feature allows the system to tailor itself to the user. Second, CID includes a mechanism that allows presentation of referents that are most likely to be useful, thus providing focus for the search in a hypertext database. (The user can, however, access all the descriptors and referents at any time). This model consequently improves the performance of the system. This paper is structured as follows. In the second section, the problem of browsing through large documents is reviewed. The

[1]Dr. Mathé is currently under contract 3004-47-2 to Recom Technologies, Inc.

third section presents the current CID system, and the fourth section describes how CID's subsystem IARC (Index Acquisition and Refinement according to Context) is able to acquire new knowledge. In the fifth section, some of the lessons learned from analyses of existing technical documentation systems, and from the design, implementation and the first tests of CID are given. Finally, the sixth section presents related work, and the seventh section discusses our system's limitations and our plans for future research issues.

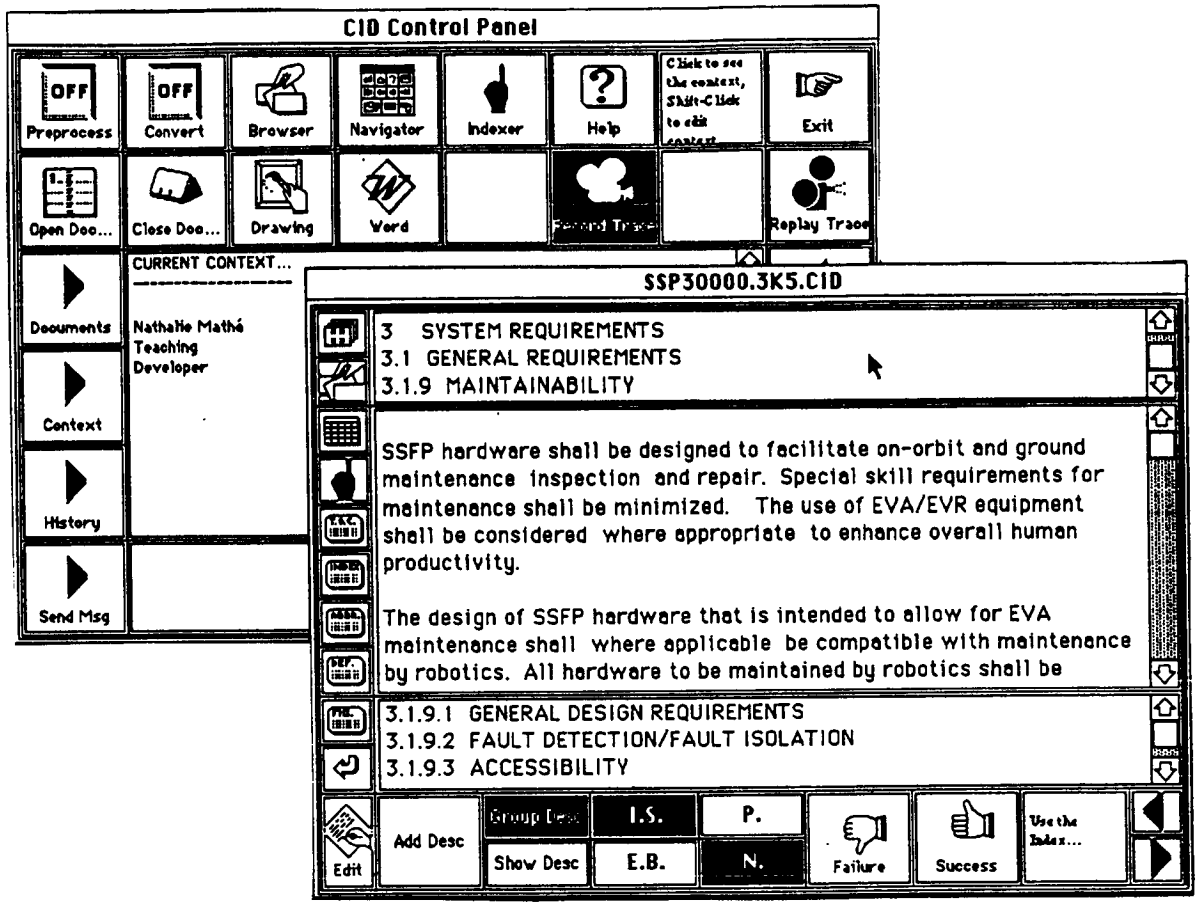


Figure 1. Part of a typical CID screen. In the lower-right window, the upper field includes the hierarchy of the displayed referent, the middle field is the actual text and the bottom field includes the next referents in the hierarchy. The upper-left window is the CID control panel.

THE PROBLEM

From analyses we have conducted during the design (and redesign) of CID, and from the current users' feedback, the following issues have become clear. When looking for specific information, people usually employ both the table of contents (hierarchical browsing) and the index of the available documents to guide their search. Even then, however, the search is not very focused, and it is common for users to examine several places in the documentation before finding the information needed. For example, when the index is used and a descriptor is looked at for the first time, the first page indicated is usually chosen. If the corresponding information turns out to be relevant for the user's purposes, the user is satisfied, and the retrieval process is finished. If, on the other hand, the corresponding information is not relevant, the user will normally go back to the index and try the second page proposed, and so on. This leads to a sequential trial-and-error decision process, which can be tedious and slow. To avoid repeating this lengthy process each time, people tend to build context around the descriptors already used. For instance, they tend to

remember that a particular referent was successful in a specific situation. This memory however does not generally last very long unless the same context occurs often.

CID has been designed to assist in this process by attempting to immediately provide the user with the information that is likely to be relevant to him/her *in context*. This paper presents an approach *to indexing in context*, in which indices are particular procedures that are modified incrementally *by experimentation*, remembering what the user found useful in a specific context. Using this approach, CID can provide tailored guidance to users browsing a document, by watching the users perform their browsing tasks. As Chen and Dhar [4] mention, to make large information banks more accessible by computer, it is best first to try and understand how reference librarians actually help users, and then to try to include these capabilities in on-line systems. The major problem in incorporating a model of user-librarian interactions into the system is the difficulty of acquiring the information for this model. CID's on-line knowledge acquisition approach allows semi-automatic acquisition of a librarian model.

Organization of Technical Documentations

In existing technical documentations, such as those common at NASA, information is structured hierarchically. Designing a complex system like the Space Station Freedom is an iterative process. Its documentation system is designed to handle a huge amount of information. It is organized around the Program Requirement Document (PRD), which establishes the highest level requirements associated with the Space Station Program. Generally, the other documents expand upon the topics expressed in the PRD. Each document includes the following major nodes: a description of the document, a preface, a table of contents, a body of text segmented into sections and subsections, an abbreviations table, a definitions table, and appendices. Each major node may contain a hierarchy of nodes. For instance, in the body of text, there is a hierarchy of sections. There are links between sections which are linear (section to next section) and non linear (reference to a section other than the next one). There are references to other major nodes within a document and sometimes to other documents. A *descriptor* [2] is any word, phrase, or piece of graphics which provides a meaningful "starting point" for a search in the documentation. A *referent* is the address of any part of text or graphics. Descriptors are organized hierarchically. In the current implementation, a referent is typically a document section. In this paper, a referent will also represent its content. In a regular book, the table of contents provides a list of descriptors (section titles). In this case, each descriptor corresponds to only one referent (a page number). The index also provides a list of descriptors, each of which usually has several referents, i.e., a sequence of page numbers. As a result, the user may need to look at several referents before finding the information needed.

Browsing through large texts

It takes years of training to be a flight controller in the Space Shuttle Mission Control Center. As part of this training, people learn to use a large corpus of documentation to solve problems. They develop a deep knowledge of the organization of these manuals in order to access the proper sections as quickly as possible. Currently the operational documentation used by flight controllers is paper-based. In the short term, the goal of CID is to help people access documentation on a computer more efficiently. One thing CID attempts to do is to help narrow the search through documents while allowing the full browsing freedom to which people are accustomed.

People typically use descriptors to retrieve information. As they start looking for information, they normally employ the *explicit* descriptors provided in the documents, e.g., table of contents or index. Unfortunately the descriptors provided are *context-free*, and, generally each descriptor can describe many referents, the problem of decidability introduces a major problem of backtracking that the user may not accept, especially when dealing with real-time operations. As a result, people

[2]The concept of a descriptor is very important in information retrieval [15]. Building such descriptors requires expertise in the domain of investigation. We are using a technique developed by Mark Zimmerman [20] that allows full-text extraction of words associated with their frequency in the text.

usually build implicit descriptors as they browse through documents, that is they build a cognitive representation of the documentation that provides relations between pieces of information and their approximate locations in the documentation. They remember that this particular piece of information was (or was not) very interesting in a special context. These cognitive maps are later used to guide their browsing task. They are thus context- and user-dependent.

Hypertext provides good support for browsing in documentation and for naturally building associative links between descriptors and referents. To automatically help the user, the problem thus becomes to "contextify" the links between documentation nodes, that is to provide relations between descriptors and referents that are valid in the current context. These relations will vary depending on the situation and the user. Providing a context for the referents reduces the number of possible referents for a descriptor that a user has to look at and thus narrows the search. In this paper, we present a technique to acquire context for these relations automatically. We define the context acquisition problem as reinforcement of current actions, as well as discovery of "abnormal conditions" and generation of recovery actions. Our system observes the user's actions during the browsing tasks, and, by noting whether a specific referent was considered a success or failure by the user in a particular context, it is able to refine the indices to reflect their context of use. In this way, our system automatically acquires the knowledge necessary to operationalize a user model: which indices are appropriate when, and for which user. This is a significant departure from current work in user modeling, where systems are able to obtain and refine user models [10, 12], but the way these models are exploited is *hardcoded* in the systems and thus inflexible [14].

AN ADAPTIVE DOCUMENTATION BROWSING SYSTEM

A browsing facility has been developed to help users search for specific information in the available documentation. CID, like the printed documentation, includes a table of contents and an index. When the user selects a descriptor, a menu of ordered referents pops up. These referents have been found successfully in the *same context* in past retrievals. The order of referents is based on the past success rate of each referent in this context. These referents can be very different among users and in various contexts.

CID has two major components: a hypertext system and a knowledge-based system. One of the major goals of this project was to keep documentation independent from the knowledge of how to use it. This latter knowledge is represented in the system in the form of *contextual links* containing preconditions (*triggering conditions* and *contextual conditions*) and a list of referents together with an indication of how often each referent was successful (a *reinforcement slot*), and a description of the situations in which the referent was *not* successful (called an *abnormal condition*).[3] An example of a contextual link is shown in Figure 2.

```
(TRIGG. COND.   (Descriptor-1))
(CONTEXT        (C1 C2 C3))
(ACTIONS        (R1 +5 ((AC1 1) (AC2 3)))
                (R2 +3)
                (R3 +2)
                (R4 +1))
```

Figure 2. Example of a contextual link. Descriptor-1 is valid if the context conditions (C1, C2, C3) are satisfied. The referent R1 has a *reinforcement slot* of +5, indicating that it has been successful 5 times and has been not found useful in two abnormal conditions AC1 -- with its

[3]The reinforcement slot and the abnormal condition can be seen as an indication of whether or not the user's *goal* for finding specific information was achieved.

reinforcement slot of 1 -- and AC2 -- with its reinforcement slot of 3. The referent R4 has been successful once.

Typically, in CID, a triggering condition is characterized by a descriptor. The selection by the user of a descriptor thus triggers a list of actions to be considered. When there are several descriptors in a contextual link, these descriptors are aliases or synonyms. The contextual conditions indicate under which conditions the actions presented in the contextual link are actually appropriate. Contextual conditions characterize the environment in which the retrieval has been made successfully. For instance, let us assume that one wants to retrieve some very specific information on the air conditioning in the main cabin of the Space Station. The first thing one may try is to browse the documentation with the descriptor "air conditioning." If the retrieval context can be specified, e.g., "you are a designer, you are interested in the connection of the air conditioning system, and you have very little information about the electrical circuitry in the cabin," then a more efficient search can be accomplished. The search will not be the same under another context, e.g., "you are an astronaut, you are in the Space Station, and you are freezing." Importantly, in our system, contextual conditions for a particular contextual link are learned *through experimentation.* Contextual conditions allow clustering of contextual links, and, when the system is operational, pruning of inappropriate contextual links.

The current context is set up at the beginning of the session. It can be a default profile attached to the user name and automatically set by the system after the login. It can be changed at any time by the user, or modified by the system following changes of sensor values if the documentation system is connected to a real-time system (e.g., in the case of documentation used in process control). In a given context, the user generally selects a descriptor to get a list of potential referents. When the user selects a referent R from this list, the system automatically activates the link between the selected descriptor and the referent R. This activation leads to the presentation of the referent R.

CID has two modes of operation, which correspond to the two modes of activity in documentation use: (a) *experimental browsing*: a casual approach, often seen in activities such as exploratory learning, in which the computer can take an active role by suggesting interesting information to be examined; (b) *intentional search*: a deliberate search for information to fill a particular need, e.g., to prepare a report or answer a specific question. Experimental browsing allows augmentation of the initial set of links between descriptors and referents.[4] Intentional search is used to refine existing contextual links of knowledge, by acquiring more contextual conditions or refining the existing ones. In the next section, we focus on this mode.

ACQUIRING INDICES BY EXPERIMENTATION

Following the above example, with the descriptor Descriptor-1 (triggering precondition) "air conditioning" and the current context (C1, C2, C3) "you are a designer, you are interested in the connection of the air conditioning system, and you have very little information about the electrical circuitry in the cabin." A contextual link is first triggered by the descriptor. As there are four possible referents in the documentation, the four referents, (R1, R2, R3, R4), are presented: "a list of the vendors of air conditioning systems," "a description of the air conditioning system," "a checklist of what to do when the air conditioning fails," and "a diagram of the electrical circuitry in the main cabin of the Space Station." The first one is not satisfactory in the current context: it is a failure. This is indicated by a mouse click from the user. The second and the third are also failures. Failure cases will be presented in the third paragraph of this section. Fortunately, the fourth one will give the information that is needed: it is a success. The system thus learns that R4 was

[4]Initial links can be built automatically assuming that descriptors are explicitly included in referents. Our system scans the hypertext database and extracts each descriptor together with a list of referents that corresponds to all the locations where this descriptor has been found. The corresponding contextual links generated this way are context-free. However, this automatic approach is generally not sufficient because some referents cannot be implicitly described by a descriptor included in the text. In this case, human intervention is necessary, i.e., the user can generate his/her own descriptors associated to particular referents.

successful in this context. At the next retrieval under the same context (C1, C2, C3), R4 will be presented automatically from Descriptor-1.

To observe the user, inputs to CID include user judgments on the success of actual retrievals. After a referent has been found, the user can select either "success" or "failure." The system automatically records this selection by adding +1 or -1 to the reinforcement slot attached to the original contextual link referent inferred by the descriptor used.

Suppose now that, using the same contextual link, a particular situation is observed in which the user indicates R1 as a failure. It would now be inappropriate to repeat this experience again and again. Instead, CID notes that an *abnormal condition* has been encountered, and this knowledge is added to the contextual link. Like actions, reinforcement values are also associated with abnormal conditions. Abnormal conditions can be seen as exceptions to the "normal" use of the contextual link. When a failure occurs, the system attempts to obtain from the user the reason for this failure. A list of previously acquired abnormal conditions is automatically presented to the user when he/she indicates his/her willingness to provide an explanation. The user may select one of the explanations provided or generate a new one. The selection is then processed automatically and kept in the corresponding contextual link as an abnormal condition.

If the abnormal condition is observed many times, its negation will automatically be added into the appropriate context for the contextual link, as the situation which used to be considered "abnormal" can be considered as "normal." A rote learning algorithm used to augment the system's knowledge has been presented in [1].

LESSONS LEARNED AND TESTS IN PROGRESS

We have analyzed [3] various uses of documentation systems available at NASA including Space Station Freedom (SSF) program requirements documents, and Space Shuttle operations procedures manuals. The former type of documentation has been called design documentation, and the latter operational documentation.

Design documentation is generally handled using keyword search. People find this very difficult in practice because keywords are used in a full-text search mode. Consequently, people using such systems come up with either hundreds of references or nothing, according to the recall/precision criterion [15]. We have found that CID's approach introducing the experimental browsing mode, allows the user to index referents with concepts that are not necessarily words or term-phrases included in the text. Another aspect is that current systems used at NASA have poor navigation capabilities (often none at all). People tend to construct their own cognitive maps of the documentation even if nothing is provided to make explicit the documentation topography. We have found that explicit maps of the documentation are very useful. These maps can be local ("where to go next?"), or global ("where am I?"). They can also present either the hierarchical structure of the documentation (local or global tables of contents), or the conceptual relationships between referents via descriptors (local or global conceptual indexes). An example of use of a contextual link as a local aid is provided in Figure 3.

Operational documentation systems (usually paper-based) are generally handled using tables of contents. Furthermore, expert users tend to develop robust search strategies based on experience and context. We have observed that, unlike the design documentation users, operational documentation users have very integrated cognitive representations of the documentation they use, i.e., they know its hierarchical structure and an extensive set of conceptual links. The reason is that operations people, in a Mission Control room for instance, are highly trained to solve problems using operational manuals. Furthermore, they update such documentation from their own experience (not only its indexing but also its content). CID will facilitate this painful and expensive process.

We are currently testing CID with SSF and Space Shuttle specialists in order to get more information on the level of acceptance of CID learning capabilities and behaviors by both design and operations people. Initial results indicates that CID is very useful both for indexing documents

in context, and for improving the revision process of the documentation itself. Context-sensitive information retrieval gives extended possibilities such as providing search expertise from other users, e.g., "what would John Smith do in this situation?"

The current version of CID is implemented on a Macintosh II Cx with HyperCard (version 2.1) and uses external functions in C. HyperCard is a good prototyping tool that is easily disseminated at NASA because of its widespread availability. This allows CID to be tested on a very large scale. Although we do not have quantitative results yet, our experience with CID to-date is very encouraging. The Space Station Freedom documentation application currently contains several Mb of text and hundreds of descriptors linked to the documentation.

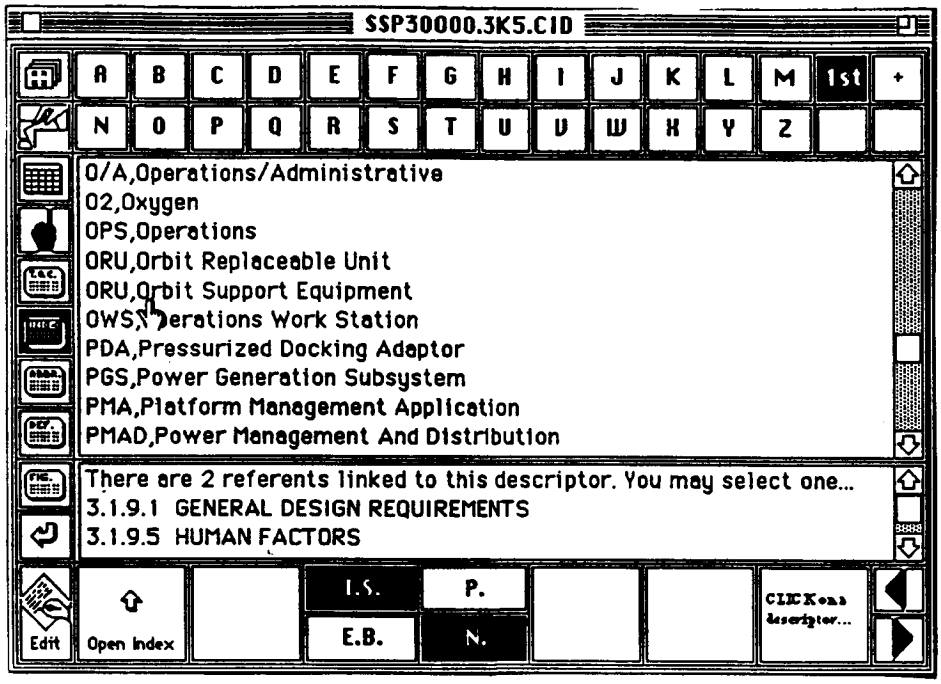


Figure 3. Use of an index in context. Here, the context was characterized by the type of user and the type of task. The user clicked on the descriptor "Orbital Replacement Unit". CID suggested two referents in the current context.

RELATION TO OTHER WORK

Semantic indexing has been investigated by several authors. Dumais et al. [7] propose a method for organizing nodes into a semantic structure on the basis of the overlap of the descriptors used in the referents. Stotts and Furuta [16] proposed a model of hypertext based on Petri nets. Their system enforces browsing restrictions, e.g., deactivates some links. Like CID, a medical handbook system described by Frisse and Cousins [8] separates the "index space" from the "document space." They have shown how some index architectures can be exploited for enhanced information retrieval, query refinement, and automated reasoning. Their index space model is based on inference using belief networks of descriptors. The I^{3}R system [5] uses a Bayesian inference network to acquire information about user's needs and domain knowledge. Crouch et al. [6] have used cluster hierarchies to help navigate in hypertext structure. All these contributions to information retrieval developed methods for refining links between descriptors and referents. However, the concept of context has not been presented explicitly in any of them.

Weyer [17] advocates the fact that information should be adaptable to the learner's preferences, and links should depend on the user's previous actions and current goals. This point of view supports our knowledge-based approach to hypertext. Other approaches have been developed to acquire and refine links on-line. For instance, Kibby and Mayes [11], in their StrathTutor hypertext system try to eliminate the need for exclusively manual methods for creating links between hypertext nodes by generating links based on knowledge acquired when the user browses through the system. Also, Monk presents a method for constructing a personal browser [13]. In this approach, the system monitors the user's navigation behavior and interrupts the user to ask whether it should add a node to the browser when it has been accessed frequently.

CID uses the concepts of context and abnormal conditions to learn from users. The work done on exceptions by Winston [19] and Williamson [18] is similar to our use of abnormal conditions. We have extended this approach with the use of dynamic reinforcement, and have incorporated these theoretical considerations in an actual implementation.

Finally, this work is a departure from current work on user modeling research, in which user models (possibly acquired automatically) are exploited by the system in predefined ways. As a result, these systems cannot update their user models based on experience. In contrast, our system can perform this updating automatically. In other words, our systems learns to *operationalize* user models automatically, based on experience.

CONCLUSION AND FUTURE DIRECTIONS

Current results have shown that hypertext is a good programming tool for the development of documentation systems. While hypertext systems increase accessibility, they do not provide any built-in selectivity mechanism. In other words, while non-linear or hypertext systems may dramatically increase the accessibility of information, this increased accessibility may magnify an already severe problem of selection [9]. For these reasons, our knowledge-based system technology can be very helpful in alleviating the selection problem and cognitive overhead of the user. Our approach to retrieval is unique because the design of contextual links to retrieve information is based not only on the way the documentation has been built, but also on user's information requirements and suggestions when they are operating systems. Thus, the user continually augments and refines the intelligence of the retrieval system.

Besides providing an intelligent interface for browsing large documents, the ability of our system to automatically acquire the context in which strategies are appropriate is significant. First, it allows the system to provide a *tailorable* browsing facility. Indeed, the system will learn which referents are to be presented for which user. Second, it shows that it is feasible to immediately incorporate the user's feedback into the system's knowledge, with the possibility of improving the system's performance. In this way, there is no need to collect and analyze large amounts of information about how users interact with the system because the system performs this task itself.

Many issues remain to be addressed. We briefly describe some of these here. First, formalization of the contextual conditions is still problematic. Contextual conditions should be minimal to avoid excessive calculations, but they must include as much information as possible to characterize the current situation. To solve this problem, future work will include the development of a context clustering mechanism. Second, we are extending CID to incorporate dynamic context, deduced from user's actions or associated to a dynamic environment. Third, we are developing a semantic similarity measure between referents, to let the user access information from a set of descriptors. (rather than a single descriptor). Finally, we are developing and evaluating a graphical contextual links browser to help navigate in large CID documents.

ACKNOWLEDGMENTS

I would like to thanks Celeste Plaisance, Mark Gersh and Kerry Soileau for their active support in evaluating CID with users at NASA HQ and at Mission Control, JSC. Thanks to Bharathi Rhagavan and Josh Rabinowitz for their contribution to CID implementation.

REFERENCES

[1] Boy, G.A., "Acquiring and refining indices according to context," Proceedings of the Fifth AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, November 1990.

[2] Boy, G.A., "Computer Integrated Documentation," MIT Conference on The Social Creation of Knowledge: Multimedia and Information Technologies in the University, held at MIT, April 6, 1991.

[3] Boy, G.A., "Computer Integrated Documentation," Technical Memorandum 103870, NASA Ames Research Center, Moffett Field, CA, September, 1991.

[4] Chen, H. and Dhar, V., "Reducing indeterminism in consultation: A cognitive model of user/librarian interactions," Proceedings of the Sixth National Conference on Artificial Intelligence, Seattle, Washington, July 1987.

[5] Croft, W.B. and Turtle, H., "A Retrieval Model for Incorporating Hypertext Links," Hypertext'89 Proceedings, pp. 213-224, ACM press, New York, 1989.

[6] Crouch, D.B., Crouch, C.J. & Andreas, G., "The Use of Cluster Hierarchies in Hypertext Information Retrieval," Hypertext'89 Proceedings, pp. 225-237, ACM press, New York, 1989.

[7] Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S. and Harshman, R., "Using Latent Semantic Indexing to Improve Access to Textual Information," Proceedings of the ACM CHI'88, pp. 281-285, Washington D.C., May 15-19, 1988.

[8] Frisse, M.E. and Cousins, S.E., "Information Retrieval from Hypertext: Update on the Dynamic Medical Handbook Project," Hypertext'89 Proceedings, pp. 199-212, ACM press, New York, 1989.

[9] Jones, W.P., "How do we distinguish the hyper from the hype in non-linear text ?" Proceedings of INTERACT'87, Elsevier Science Publisher, Holland, 1987.

[10] Kass, R. and Finin, T., "Rules for the implicit acquisition of knowledge about the user," Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 295-300, Seattle, Washington, July 1987.

[11] Kibby, M.R. & Mayes, J.T., "Towards Intelligent Hypertext," Hypertext Theory into Practice, McAleese, R. (Ed.), Albex, 1989, pp. 164-172.

[12] Kobsa, A., "A taxonomy of beliefs and goals for user models in dialog systems," In User Models in Dialog Systems, Kobsa, A. & Wahlster,W. (Eds.), Springer Verlag, Symbolic Computation Series, Berlin Heidelberg New York Tokyo, 1989.

[13] Monk, A., "The personal browser: A tool for directed navigation in hypertext systems," Interacting with Computers, 1, 2, 1989, pp. 190-196.

[14] Paris, C.L., "The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise," PhD thesis, Columbia University Department of Computer Science, 1987.

[15] Salton, G., "Automatic Text Processing: the transformation analysis, and retrieval of information by computers," Addison Wesley, Redding, Ma, 1989.

[16] Stotts, P.D. and Futura, R., "Adding Browsing Semantics to the Hypertext Model," Proceedings of the ACM Conference on Document Processing Systems, pp. 43-50, Santa Fe, NM, December 5-9, 1988.

[17] Weyer, S.A., "As we May Learn," In Interactive Multimedia: Visions of Multimedia for Developers, Educators, & Information Providers, Aubron, S. & Hooper, K. (Eds.), Microsoft Press, 1988, pp. 87-103.

[18] Williamson, K.E., "Learning from exceptions in databases," Machine Learning: A Guide to Current Research, Mitchell, T.M., Carbonell, J.G. & Michalski, R.S. (Eds.), Kluwer Academic Publishers, 1986.

[19] Winston, P.H., "Learning by augmenting rules and accumulating sensors," Proceedings of the International Machine Learning Workshop, pp. 22-24, Monticello, Illinois, June 1983.

[20] Zimmerman, M., "TEX version 0.5," Technical Report, Silver Spring, MD., 1988.

INFORMATION FOR THE USER IN DESIGN OF INTELLIGENT SYSTEMS

Jane T. Malin
NASA Johnson Space Center
Intelligent Systems Branch - ER22
Houston, TX 77058
(713) 483-2046

Debra L. Schreckenghost
The MITRE Corporation
1120 NASA Road One
Houston, TX 77058
(713) 333-0944

ABSTRACT

Recommendations are made for improving intelligent system reliability and usability based on the use of information requirements in system development. Information requirements define the task-relevant *messages* exchanged between the intelligent system and the user by means of the user interface *medium*. Thus, these requirements affect the design of both the intelligent system and its user interface. Many difficulties that users have interacting with intelligent systems are caused by information problems. These information problems result from (1) not providing the right information to support domain tasks, and (2) not recognizing that using an intelligent system introduces new user supervisory tasks that require new types of information. These problems are especially prevalent in intelligent systems used for real-time space operations, where data problems and unexpected situations are common. Information problems can be solved by deriving information requirements from a description of user tasks. Using information requirements embeds human-computer interaction design into intelligent system prototyping, resulting in intelligent systems that are more robust and easier to use.

INTRODUCTION

Many difficulties that users have interacting with intelligent systems are caused by information problems. These problems are especially prevalent in systems used for real-time operations, where timing constraints make it essential that intelligent systems communicate effectively with their users (users in space operations are called *operators*). The following example illustrates a typical information problem in this environment.

> *Example*: A user's task is to detect event Y, which occurs when sensor A is bad and switch B is off. The intelligent system displays the current status of sensor A and state of switch B. If a change in the displayed value from either sensor A or switch B occurs before the user looks at the display, the user misses event Y.

On the surface, the problem with this system appears to be caused by "bad" user interface design (i.e., overwriting the display of data from sensor A and switch B before event Y can be detected). A closer look, however, reveals that the so-called bad user interface is merely a symptom of an underlying information problem (i.e., the information of interest is event Y, but the intelligent system does not provide that information).

This paper characterizes the information problems encountered when building intelligent systems for real-time space operations, and makes design recommendations for solving these problems. These results are based on experience gained in designing intelligent systems for space operations at the National Aeronautics and Space Administration (NASA). The authors have extended their design experience with case studies of intelligent systems built and used at NASA (Malin, et al., 1991). They have also collaborated with Woods and Potter (Woods, et al., 1991; Potter and Woods, 1991) concerning new user interface designs addressing some of these information problems.

This paper was written to assist intelligent system designers in designing for more effective communication with users, and to inform intelligent system tool builders and human factors engineers of ways to better support these system designers. The first section describes the information problems encountered in real-time space operations. The next section introduces the concept of information requirements as an approach to handling these information problems. The third section discusses the design of intelligent systems for effective communication with users. This includes describing the information needed to monitor the domain system and supervise the intelligent system, and proposing alternatives to typical user interface design approaches. The final section summarizes how these recommendations improve intelligent system reliability and usability. The topics discussed in this paper are covered in greater detail in a NASA Technical Memorandum (Malin and Schreckenghost, 1992).

INFORMATION PROBLEMS IN REAL-TIME SPACE OPERATIONS

A key observation from the case studies is that many perceived user interface problems in intelligent systems are actually information problems. These information problems result from (1) not providing the right

information to support *domain tasks*, and (2) not recognizing that an intelligent system introduces *new user supervisory tasks* that require new types of information. These problems are made more difficult by failure to consider how intelligent systems operate in space environments (e.g., effect of data quality and availability on intelligent system behavior) and failure to integrate intelligent system operations with other operations. If not solved, these information problems impact intelligent system reliability and usability.

The first information problem is failure to provide the right information to support domain tasks. The most common tasks performed by intelligent systems being built today are fault monitoring, detection, and diagnosis of cause. The information needed to perform these tasks includes the important behaviors, interesting relationships, and significant changes occurring in the domain system, i.e., *monitored process* (Woods, et al., 1991). To interpret this information, the operator must also understand the behavior expected to occur, and the thresholds delimiting significant or interesting changes (i.e., transition points). For example, many of the operator's decisions during fault management require information about functional capability (what functionality has been lost, how the mission is impacted by that loss). Yet the information typically communicated by the intelligent system consists of device failures shown on schematics or listed in message logs. Such communication does not support the operator in identifying the important changes (e.g., lost functionality) and relationships (e.g, how failures impact mission goals). Considerable effort is required to use this information to make fault management decisions. Thus, common practice in communicating with the operator does not provide the information needed to make fault management decisions.

The second information problem is failure to design for user supervision. New user supervisory tasks include both monitoring ongoing intelligent system activities, and guiding and correcting the system when it malfunctions. Intelligent systems are usually not designed to be managed because it is not well recognized that they need to be managed. This omission in design arises from two misconceptions: (1) that the intelligent system is more knowledgeable than its user, and (2) that the intelligent system can be designed to prevent all errors from occurring. In fact, the typical space operations user (a flight controller) is also a domain expert. This expert user is more knowledgeable than the intelligent system and is well qualified to supervise it. The misconception that all intelligent system errors can be prevented results from unrealistic assumptions about the space operations environment and how intelligent systems operate within that environment. Due to the complexity of this environment, the behavior of the monitored process cannot always be accurately predicted, and unexpected situations occur. Because they are unexpected, the knowledge base does not address them and the intelligent system can respond anomalously. Additionally, data problems (e.g., stale, noisy, or biased data) are common in space environments and can cause intelligent system error.

Although intelligent systems can be designed for supervision and correction (Land et al., 1992), they are not typically designed that way. Often, the intelligent system provides no means for the user to respond to system errors, apart from turning the system off. If not designed for user supervision, the intelligent system can also be difficult to understand (what Abbott calls a "magical" system; Abbott, 1991) because it doesn't provide the user with necessary information about system processing. The human supervisor must understand what the intelligent system can do (its capabilities), what it is currently doing (its activities), and why (its reasoning strategies). Without such an understanding, the supervisor cannot guide and correct it. Providing this additional information to supervise the intelligent system can overload the user, however, if it is not effectively managed. Because the intelligent system is embedded in a larger support system, information from that larger system can be used when compensating for intelligent system errors (e.g., operator can use that information to take over intelligent system tasks).

INFORMATION REQUIREMENTS FOR SYSTEM DESIGN

The information problems in real-time space operations that were discussed in the previous section can be characterized as not providing the right information at the right time to support fault management tasks for the monitored process and user supervisory tasks for the intelligent system. An understanding of both of these types of tasks is necessary to determine what the "right" information is and when it should be provided. Using a description of these tasks, the designer can define the task-relevant messages exchanged between the intelligent system and the user by means of the user interface medium (i.e., the *information requirements*). These information requirements are then used in designing the system. Because they are based on a description of how the user will interact with the intelligent system, information requirements include operational considerations early in system design.

Information requirements affect the design of both the intelligent system (i.e., what information to represent) and its user interface (i.e., what information to present). Thus, intelligent system design and user interface design are not independent efforts, but aspects of a single development process. Considering human-computer interaction (HCI) as part of system development integrates user interface design into overall system design. Since information requirements affect both intelligent system and user interface design, HCI expertise is needed throughout the development of the

system. Early in system development, HCI expertise is needed for describing task-level information requirements. Knowledge of display and control software and hardware is needed to design media for presenting this information, which may include early development of user interface design concepts as prototypes or storyboards. This approach also necessarily involves users early in system design to describe the task and assist in identifying information requirements.

A task-based approach to requirements definition solves many of the information problems described earlier. Identifying information requirements does not require a full task analysis, such as the GOMS analysis (Kieras, 1988). Only the high-level monitoring, control, and decision-making tasks for managing the monitored process and supervising the intelligent system need be identified (see next section for an illustration). Finer-grained task analysis techniques are not appropriate for this purpose, because they are designed for detailed user interface design at the dialogue or display level.

There is much yet to be learned about how to develop high-level task descriptions for the purpose of identifying information requirements. Formal methods will most likely evolve from approaches proposed by Rasmussen (1986) and Mitchell and Miller (1986), and from distributed agent communication approaches from artificial intelligence. In the interim, information requirements can be identified by developing and informally analyzing scenarios. Such scenarios would include the identified joint human-computer tasks, and would represent managing both the monitored process and the intelligent system. These scenarios are evaluated to identify the information that must be exchanged between the user and the intelligent system. Alternative task allocations can also be evaluated for ability to recover from intelligent system errors, to accommodate changes in task priority or workload, and to coordinate human and intelligent system activities. Both prototypes and storyboards can be used to evaluate operational scenarios.

Information requirements are the basis for selecting what information should be represented and presented for a task. This guarantees that all the needed information, and only the needed information, is provided. This solves the information problems related to magical systems and information overload. Additionally, information requirements provide a more objective and rigorous basis for evaluating a design than the usual approach in which a design is good if the users like it (what Abbott calls design by "Mikey likes it";Abbott, 1992).

COMMUNICATION BETWEEN HUMAN AND INTELLIGENT SYSTEM

Effective human-computer communication requires striking a balance. On one hand, the intelligent system must provide enough information for the user to understand its behavior (i.e., avoiding a magical system). On the other hand, the intelligent system must not provide so much information that it interrupts or distracts the user from more important tasks. The user is already overloaded with information. The problem of information overload becomes especially important in real-time environments where complex, high risk tasks are performed, and where human errors can represent serious risk.

Most of the intelligent systems studied communicate with the user in at least one of the following ways:

- *message list*: a chronological list of state and status assessments and/or action recommendations
- *annotated schematic*: a graphic representation of the physical structure of the system, annotated with sensor measurements or state/status assessments
- *explanation*: a conversational style of providing justification for an intelligent system conclusion

These typical approaches to communication are not effective at achieving balanced communication. They often do not represent the right type of information for user tasks or do not present it in a way effectively supporting real-time operations.

Balanced communication is achieved by developing a shared understanding of the ongoing situation, so that the intelligent system and user make decisions based on the same information. Such an understanding is developed over time by monitoring the same information, including the environmental events, the actions of the crew and flight controllers, and the behavior of both the monitored process and the intelligent system in response to those events and actions. Understanding the behavior of a system and its capabilities to respond to a situation requires having some "visibility" into the system. Providing visibility is providing an *unobstructed view* to the user. To be *unobstructed*, nothing should get in the way of sight (i.e., information is clearly presented, with the relevant information apparent). A *view* includes a perspective. Information for visibility into the monitored process is represented from the perspective of managing process operations and failures. Information for visibility into the intelligent system is represented from the perspective of coordinating with and managing operations and errors in this system. This section describes the types of information needed for both of these perspectives, and discusses alternatives to the traditional forms of communication.

Information for Managing the Monitored Process

Managing the monitored process requires that the operator monitor process behavior for anomalies and respond to those anomalies. To detect anomalies, the operator must have some expectation of what process behavior should be (i.e., nominal behavior). Typically, an alarm is annunciated when behavior is not within the limits defining nominal behavior. Multiple alarms may be issued shortly after the initial alarm due to failure propagation into other systems and redundant alarms. Additionally, anomalies can have serious implications for safety and mission objectives, and can impose hard timing constraints. Thus, managing information from multiple alarms increases workload just at the time when the operator can least afford it.

Responding to anomalies in space systems is a complex decision-making process, often with high stakes (e.g., potential loss of crew, failure of costly experiment). The operator must make the following decisions when an anomaly occurs:

- Determine if any action is required in response to the anomaly
- Distinguish failures from false alarms
- If more than one response is possible, select one
- When action is taken, evaluate if it is effective

A significant amount of information is needed to make these decisions. The cause of the anomaly (or a set of possible causes) must be identified. The impacts of the anomaly must be determined, including the immediate consequences to mission objectives and safety (e.g., lost functionality) and the potential for future consequences (e.g., failure propagation potential). Response options must be delineated and the "best" response enacted. The effect of response procedures must also be monitored for adverse or unexpected effects.

To support alarm management and anomaly response, the intelligent system should provide information that improves the operator's understanding of the situation. This requires focusing the operator's attention on what is diagnostically important, and quickly and clearly indicating the diagnostic content and relationships in this information. Especially when the system supports operations that change process states, it is important to call attention to events (e.g., state transitions) and procedure-driven activities (e.g., configuration changes) preceding the anomaly, as context for interpreting an anomaly. The example in Figure 1 illustrates how knowledge of preceding events can assist the operator in managing the monitored process.

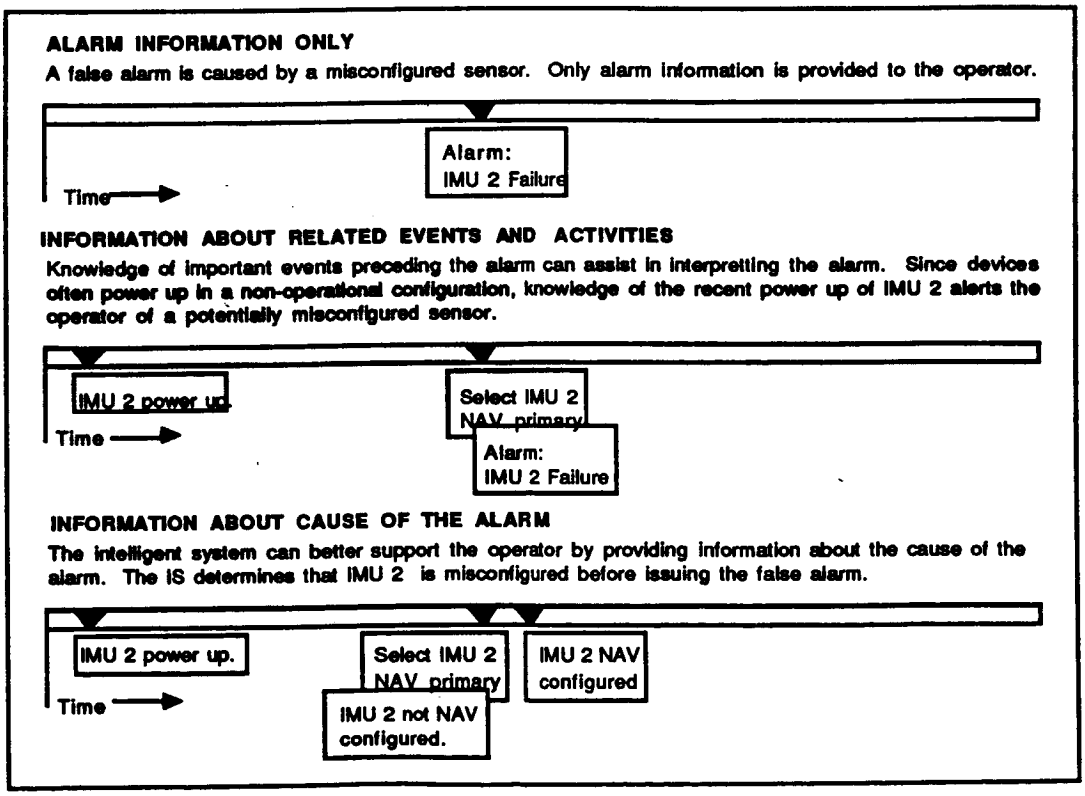


Figure 1 - Using Knowledge of Preceding Events to Improve Operator Understanding of Situation

Both message lists and schematics can obscure intelligent system information important to the operator (Woods, et al, 1991). As shown in the example, situations develop as patterns of events indicated by changing state and status. Schematics can only present the latest event (i.e., the current state). Additionally, if events are not related to physical structure (e.g., functional status), they can be difficult to present clearly using a schematic. Message lists do capture some event history, but do not represent the relationships between events (e.g., the temporal distance between events) necessary to reveal these patterns. Since chronology is the means of sorting messages, related information can become dissociated. Intelligent system designers should consider alternatives to message lists and schematics that assist the operator in seeing patterns of events as they occur. For example, Potter and Woods are investigating timelines as an alternative to message lists (Potter and Woods, 1991). Representation of functional information instead of physical information can be effective for supporting anomaly response (Malin et al., 1991).

Information for Supervising the Intelligent System

Possibly the most significant problem in intelligent system design is failure to recognize that use of an intelligent system poses new tasks for the operator. In addition to managing the monitored process, the operator must now supervise the intelligent system, including monitoring and coordinating its activities, and responding to its errors. But intelligent system are rarely designed for supervision. The traditional means of communicating with an intelligent system (message lists, schematics, and explanation) do not support the operator in monitoring its activities and understanding its reasoning. And, when system errors occur, the operator usually has few options for responding to them (the restart button or the power plug).

Designing the intelligent system for supervision means providing adequate information for the operator to understand what it is doing and to know how to respond to its errors (i.e., providing visibility into the intelligent system). Similar to providing visibility into the monitored process, providing visibility into the intelligent system means making important system behavior evident as a situation develops (i.e., conclusions and reasoning strategies). It means showing intelligent system activities in progress and how well these activities are achieving goals. It also means informing the operator of the critical evidence used to draw a conclusion and the confidence in that conclusion (hypothesis or conclusion). This information should be presented in a way that reinforces the operator's understanding of the intelligent system's reasoning strategy (Chandrasekaran, et al., 1989). Thus, managing the intelligent system means providing a lot of new information to the operator. There is a risk of overloading the operator if the system is not carefully designed to assist the operator in performing these new tasks and managing the new information needed for these tasks.

Explanation is the common approach to providing visibility into the intelligent system. Most explanation systems operate retrospectively (like help systems), requiring the operator to wait until after a situation has stabilized (and the intelligent system has reached a conclusion) before attempting to get an explanation. In real-time support environments, the operator often cannot afford to wait until system behavior stabilizes, for the safety impacts may be too great. Such event reconstruction is also not sufficient for coordinating shared human-computer activities. Additionally, the conversational style of explanation can be distracting and can contribute to information overload.

Even if the operator had time to interrupt ongoing activities for an explanation, a problem would remain with traditional approaches to explanation. Affecting the operator's behavior requires that the operator both *understand* the meaning and consequences of the explanation, and *accept* them as correct. Most explanation systems assume that failure to influence user behavior occurs because the user does not understand the explanation, and continue to provide more detailed justification directed at improving understanding. Contrary to this assumption, acceptance does not necessarily result from understanding. The user may understand the intended meaning but choose not to believe it, due to information unknown to the intelligent system or not considered by it. Or the user may believe the information but be unwilling to alter behavior, due to the belief that adverse side-effects will result or that the consequences of altering behavior are of no significance. Typical explanation approaches ignore the better information available to expert users such as flight controllers. Explanations should provide the kind of intelligent system visibility that effectively supports real-time detection of intelligent system anomalies, and even diagnosis and formulation of responses.

Thus, alternatives to explanation should both avoid the need for retrospective dialogue and support the user's supervisory task. A promising approach is for the human and intelligent system to share information and representations. Using the same information, the user can follow operational situations and compare conclusions with intelligent system assessments, as a part of normal monitoring and control operations. A good first step is to clarify intelligent system reasoning by displaying plots or tables of critical evidence supporting its conclusions, as shown in Figure 2. This example also illustrates use of the following information to support user understanding of the

monitored process that parallels the data used for intelligent system assessments:

- Present information describing the situation as it develops, including both monitored process behavior and intelligent system activities.
- Establish expectations about what might happen next (e.g., annotate data with predictions).
- Identify critical transitions and regimes of behavior (both current and pending; e.g., process or task information to annotate displays, Forbus, 1991).

Such information both supports the operator's tasks and provides some on-the-job training, by reinforcing the operator's mental models of both the monitored process and the intelligent system.

A representation shared between agents is prevalent as the basis of communication in many domains, including humans advising robots using a shared task representation (Martin and Firby, 1991), distributed machine agents planning tasks using shared goals (Decker, et al., 1991), and designers developing shared mental models (Sycara and Lewis, 1991). Shared representations are achieved by designing the intelligent system's representation to correspond to the expert operator's representation for performing the management tasks. To achieve this, it may be necessary to make implicit task information explicit in the intelligent system (e.g., to support robot advice-taking in Martin and Firby, 1991).

The concept of shared representations supports development of common knowledge between the user and the intelligent system. If intelligent system conclusions can be represented in a way that is self-evident to the user, such an intelligent system can become self-explanatory. With such a system, the user would not need to analyze the details of intelligent system reasoning. Less attention and time would be needed to effectively supervise the intelligent system.

CONCLUSION

The information problems described in this paper can cause many intelligent system design problems. A solution to these problems has been proposed based on designing from information requirements. Design recommendations have been made for improving human-computer communication of this information. The impact of these problems on intelligent system reliability and usability is now described, and the benefits of solving these problems delineated.

Reliability is a critical design issue, since an unreliable and uncontrollable intelligent system can impact the safety of crew and space systems. Intelligent systems that do not perform reliably cannot successfully provide real-time decision support. Thus, solving those problems affecting reliability should be of first priority to the intelligent system designer. Information problems affecting system reliability (*shown in italics*) and recommendations for solving these problems are summarized below:

The design may fail to support the user in supervising intelligent system activities and recovering from intelligent system errors.

Designing the intelligent system for supervision means keeping the user informed of its conclusions, behavior, capabilities, and reasoning strategies in the context of

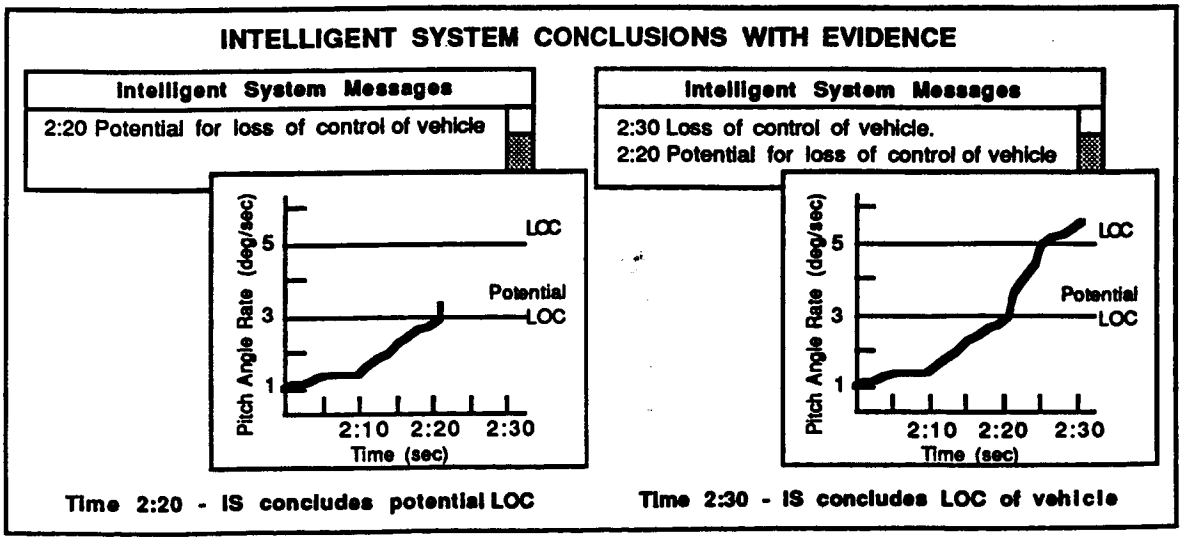


Figure 2 - Displaying Evidence Supporting Intelligent System Conclusions

ongoing events. It means providing the user with some recourse when intelligent system errors occur, such as reallocating intelligent system tasks to the user. To permit such task reallocation, information from data sources other than the intelligent system should be accessible, and independent of intelligent system conclusions.

The design may fail to handle bad or unavailable data, or unexpected situations.
Robustness to data deficiencies and unexpected situations is achieved by minimizing the bad data processed by the intelligent system and by providing capability to recover from errors. Methods include providing for data preprocessing and system self-correction (e.g., retract inconsistent conclusions), and designing for user supervision.

Intelligent system activities, reasoning strategies, and capabilities may be misunderstood and often overestimated by the user (i.e., a magical system).
To avoid building magical systems, it is necessary to provide dynamic feedback about ongoing system activities, and to reinforce the user's understanding of system capabilities. This information should be integrated into a display of the overall situation that develops over time. This permits the user to develop an understanding of system activities as they occur instead of trying to retrospectively develop such an understanding after problems occur.

Information problems also impact intelligent system usability. Intelligent systems that are difficult to use have an increased risk of user rejection and user errors. Solving problems affecting usability should improve the chances of intelligent system success. Information problems affecting system usability (*shown in italics*) and recommendations for solving these problems are summarized below:

Common practice in user interface design may increase user workload and fail to provide the important, task-relevant information (i.e., message lists, schematics, and explanation).
Deficiencies in user interface design result from misunderstanding what information is needed by the user. The user interface should be designed from a description of the task-based information requirements. This information should be presented to illustrate situations as they develop, including the behavior of both the monitored process and the intelligent system. Alternatives to explanation should clarify intelligent system conclusions and reasoning strategies, including the evidence supporting these conclusions.

The intelligent system may not be integrated with the support system.
Usually the intelligent system does not operate as a stand-alone system, but is instead embedded in a larger support system. Information from the intelligent system must be integrated with the sources of operational data, and the intelligent system displays must be integrated with other displays.

The intelligent system may not be designed for coordination with the user.
Designing for coordination requires avoiding unnecessary interruptions or interference in user activities. Changes in task allocation and dependencies between tasks (including required information exchange) represent points of coordination that constrain the design. An essential element of coordinating shared tasks is providing feedback about ongoing intelligent system activities.

Applying these recommendations improves safety and reduces cost. Building reliable intelligent systems reduces safety threats due to system error. Building usable systems reduces the potential for user error and improves user acceptance. This reduces the chance of the system not being used, and minimizes costly redesign of the system. The results are safer operations using intelligent systems and reduced cost of building these systems.

REFERENCES

Abbott, K., Internal memo, Human Automation Integration Branch, Langley Research Center, NASA, November 1991.

Abbott, K., Personal communication, Human Automation Integration Branch, Langley Research Center, NASA, 1992.

Chandrasekaran, B., Tanner, M., and Josephson, J., "Explaining Control Strategies in Problem Solving," IEEE EXPERT, spring 1989, pp. 9-24.

Decker, K. Garvey, A., Humphrey, M., and Lesser V., "Effects of Parallelism on Blackboard System Scheduling," Proc. IJCAI, Australia, August 1991.

Forbus, K., "Qualitative Physics as a Tool for Human-Computer Interaction," The Institute for the Learning Sciences, Northwestern University, Evanston, IL, 1991.

Kieras, D., "Towards a Practical GOMS Model Methodology for User Interface Design," HANDBOOK OF HUMAN-COMPUTER INTERACTION, North Holland: Elsevier, New York, 1988.

Land, S., Malin, J., and Culp, D., "DESSY: Making a Real-time Expert System Robust and Useful", Proc. Space Operations, Applications, and Research Symposium, Houston, TX, August 1992.

Malin, J., Schreckenghost, D., Woods, D., Potter, S., Johannesen, L., Holloway, M., and Forbus, K., "Making Intelligent Systems Team Players: Case Studies and Design Issues, Vol.1. Human-Computer Interaction Design; Vol.2. Fault Management System Cases," NASA TECHNICAL MEMORANDUM 104738, Johnson Space Center, Houston, TX, September 1991.

Malin, J. , and Schreckenghost, D., "Making Intelligent Systems Team Players: An Overview for Designers," NASA TECHNICAL MEMORANDUM at press, Johnson Space Center, Houston, TX, June 1992.

Martin, C., and Firby, R., "An Integrated Architecture for Planning and Learning," ACM SIGART BULLETIN, 2, 4, 1991, pp. 125-129.

Mitchell, C., and Miller, R., "A Discrete Control Model of Operator Function: A Methodology for Information Display Design," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 16, 3, May/June, 1986, pp. 343-357.

Potter, S. and Woods, D., "Event Driven Timeline Displays: Beyond Message Lists in Human-Intelligent System Interaction," Proc. IEEE International Conference on Systems, Man, and Cybernetics, Charlottesville, VA, October 1991.

Rasmussen, J., INFORMATION PROCESSING AND HUMAN-MACHINE INTERACTION: AN APPROACH TO COGNITIVE ENGINEERING, North-Holland, New York, 1986.

Sycara, K., and Lewis, C., "Cooperation of Heterogeneous Agents through the Formation of Shared Mental Models," AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents, Anaheim, CA, July 1991.

Woods, D., Potter, S., Johannesen, L., and Holloway, M., "Human Interaction with Intelligent Systems: Volume I -- Trends, Problems, New Directions," CSEL REPORT 1991-001, Cognitive Systems Engineering Lab, Ohio State University, Columbus, OH, March 1991.

Producing Approximate Answers to Database Queries *

Susan V. Vrbsky and Jane W. S. Liu

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

We have designed and implemented a query processor, called APPROXIMATE, that makes approximate answers available if part of the database is unavailable or if there is not enough time to produce an exact answer. The accuracy of the approximate answers produced improves monotonically with the amount of data retrieved to produce the result. The exact answer is produced if all of the needed data are available and query processing is allowed to continue until completion. The monotone query processing algorithm of APPROXIMATE works within the standard relational algebra framework and can be implemented on a relational database system with little change to the relational architecture. We describe here the approximation semantics of APPROXIMATE that serves as the basis for meaningful approximations of both set-valued and single-valued queries. We show how APPROXIMATE is implemented to make effective use of semantic information, provided by an object-oriented view of the database, and describe the additional overhead required by APPROXIMATE.

1. Introduction

Many factors can make it impossible for a database to produce an exact answer to a query. A network partition or a host failure can cause some needed data to become inaccessible. There may not be enough time to acquire all the locks and retrieve all the data needed to answer a query. For many applications it may be better for a database to produce an approximate answer when it is not possible to produce an exact answer [1,2]. We have designed and implemented an approximate query processor, called APPROXIMATE, that can produce approximate answers to database queries for these applications [3].

The problem of how to define and improve approximate answers has been addressed by many researchers [4-7]. Buneman, Davidson, and Watters [4] introduced the approximation semantics where approximations of a set-valued answer are defined in terms of the subsets and supersets of the exact answer. The approximations are monotonically improving as the subsets and supersets approach the exact answer, and an approximate answer produced earlier is never contradicted by an approximate

* This work was partially supported by NASA Contract No. NASA NAG 1-613, ONR Contract No. NVY N00014-92-J-1146, and AFOSR Contract No. 1-5-26932.

answer produced later. Ozsoyoglu et al. have addressed the problem of producing approximate answers to queries with time constraints [5]. Monotonically improving approximations of the exact answer, which are subsets of increasing sizes, are produced. Fragments of the data are processed at a time to produce these subsets. Motro [6] developed a system, called VAGUE, that provides approximate answers to vague queries in which the query qualifications are imprecise. The response closest to the query qualifications according to a distance function is the answer to a vague query. An intensional answer [7] provides an approximation that is derived without accessing the physical data in the database. General characteristics about the data objects in the exact answer are provided instead of the data objects themselves.

APPROXIMATE implements monotone query processing; an initial approximation of the exact answer is produced when query processing begins based on the information it maintains for this purpose. The approximate answer produced improves as more data are retrieved to answer the query according to the partial-order relation defined by the approximate relational model [8]. APPROXIMATE returns the exact answer if all of the needed data are available and if there is enough time to continue with the processing. The latest, best available approximate answer is returned if the user demands an answer before query processing is completed. In contrast, during traditional query processing, if there is not enough time or not all of the data are available, no answer is provided. APPROXIMATE assumes the information contained in the database and the query expression are precise. APPROXIMATE can be one of several query processors available to the system. It is implemented in a relational database system and requires little or no change to the underlying relational architecture.

This paper describes the semantics of approximation supported by APPROXIMATE. After approximations of query answers are defined in Sections 2 and 3, Section 4 describes how approximate answers are produced by APPROXIMATE. Section 5 discusses future directions of this work.

2. Approximate Relational Model

There is a natural way to approximate answers of set-valued queries; an exact answer E to such a query is a set of data objects. All the data objects in a subset of E certainly belong to E, and a data object in a superset of E is possibly also in E. Therefore, a meaningful approximation of any exact answer E can be defined in terms of a subset and a superset of E. Specifically, an approximation A of an exact answer E is the union of two sets of data objects: a *certain set* C, where $C \subseteq E$, and a *possible set* P, where $(C \cup P) \supseteq E$. C is the set of data objects certainly in E; it is produced from the stored data processed thus far. P is the set of data objects that may be in E. Data objects in P are produced based on the meta data, information about the stored data, maintained by the query processor. This approximation is denoted by the 2-tuple $A = (C, P)$.

Any exact answer E has many approximations. Given a set of approximations of E, a partial order relation $\geq$ for comparing them can be defined over the set as follows. One approximation $A_i = (C_i, P_i)$ is better than or equal to another $A_j = (C_j, P_j)$, denoted as $A_i \geq A_j$, if $P_i \subseteq P_j$ and $C_i \supseteq C_j$. This partially ordered set of all approximations of E is a lattice. In the lattice, $A_0 = (\varnothing, \upsilon)$ is the least element and the worst possible approximation of E, where υ is the cartesian product of all the domains in the schema of E. υ is the set of all possible data objects which could be in E. It can be generated without reading any data. The greatest element of the lattice is the best possible approximation and is E itself, which is represented by $(E, \varnothing)$.

In the traditional relational model, an exact answer E is a standard relation. An approximation of a

standard relation is called an *approximate relation.* As an example, we consider an AIRPORTS database that resides on-board an airplane. AIRPORTS contains the relation RUNWAYS (id, airport, length, obstructions). A pilot queries the database to locate all the runways with an easterly direction, with ids from 5 to 13, at airports O'Hare (ORD) and Midway (MDW) in Chicago. The exact answer to this query is the relation E shown in Figure 1(a). The relation A_1 shown in Figure 1(b) contains tuples on all runways at ORD and MDW. It gives all the tuples that are possibly in E and hence, is an approximation of E. The approximate relation A_2 shown in Figure 1(c) is another approximation of E. The first three tuples are certainly in E. They form a subset of E. The last three tuples are possibly in E. A_2 is a superset of E. It is a subset of A_1 and is a better approximation of E than A_1. The approximate relation A_3 in Figure 1(d) is another approximation of E. It is a subset of A_1 and is better than A_1, but is not comparable to A_2.

3. Approximations of Single-Valued Answers

The semantics of approximation defined by the approximate relational model can also serve as a basis for a meaningful semantics of approximation of single-valued queries. An exact answer to a single-valued query can be a single object retrieved from the database, the value of an aggregate function (such as "count"), or a value (such as "yes" or "no").

We consider the query "What is the color of car No. 20?" that exemplifies a special case of set-valued queries whose exact answer is a set of cardinality 1. The exact answer E is "maroon". Since any proper subset of E is the null set $\emptyset$, an approximate answer of E is, therefore, simply $(\emptyset, P_i)$ where

id	Airport	Length
6	MDW	7500
7	MDW	8000
9	ORD	11000
10	ORD	8000
13	ORD	10000

Figure 1(a). E: All easterly runways at ORD and MDW

id	Airport	Length
1	MDW	6000
6	MDW	7500
7	MDW	8000
9	ORD	11000
10	ORD	8000
13	ORD	10000
27	ORD	9500

Figure 1(b). A_1: All runways at ORD and MDW

id	Airport	Length
6	MDW	7500
7	MDW	8000
10	ORD	8000
9	ORD	11000
13	ORD	10000
27	ORD	9500

Figure 1(c). A_2: An approximation of E

id	Airport	Length
10	ORD	8000
13	ORD	10000
1	MDW	6000
6	MDW	7500
7	MDW	8000
9	ORD	11000

Figure 1(d). A_3: An approximation of E

the possible set P_i is a superset containing the exact answer. A possible value of P_i is {red, pink, maroon}, a set of three reddish colors. This answer improves as elements "red" and/or "pink" are deleted from P_i.

The exact answer to the query "How many cars of make Oldsmobile are available?" is the value of an aggregate function over the set O of data objects that satisfy the query constraints, such as make = Oldsmobile. In this case, the aggregate function is "count". Since the domain of the count function is a set of non-negative numbers in this case, as an approximation to the value of such an aggregate function, we can provide the values of the aggregate function defined over a certain set C and a superset $C \cup P$ of O; where (C, P) is an approximation of O. The values of the aggregate function count over the certain set and over the superset give us a superset of values, such as the range "2 to 10", or $\{2, 3, \cdots 10\}$. This approximate answer improves as more data objects are processed; the certain set increases in size and the possible set decreases in size. The counts over a bigger certain set and/or smaller possible set give us a smaller range, such as $\{4, 5, \cdots 9\}$.

Some queries require a yes or no answer, such as, "Is the number of runways at O'Hare greater than 10?". The answer is derived from the value of "count" over the set R of all runways at O'Hare. A meaningful approximation of the exact answer can be derived from an approximation (C, P) of the set R. The data objects that satisfy the query constraints, such as airport = O'Hare, are members of C. As long as the certain set in (C, P) contains 10 elements or less, the derived approximate answer remains "no". In addition to this value, we can also provide, as part of an approximate answer to such a query, a percentage value of the amount of data retrieved and processed thus far to produce (C, P), and the value of count over the current certain set C. An example is "No - 60%, number of runways ≥ 3". As more data are retrieved and processed, the percentage of data processed increases monotonically, and the value of the count function over C becomes closer to the exact value.

We can make the approximation semantics more meaningful by comparing subsets not only on the basis of their cardinalities, but also on the basis of some metric that measures the distances of their elements to the exact answer. We use such a measure of accuracy, called a *distance function*, to quantify how much better one approximate answer is than another. A distance function induces a partial-order relation over the set of all approximate answers of an exact answer E. The query processor uses this measure of accuracy to identify a good initial approximation of the exact answer when query processing starts. It also uses this measure to choose the next set of data objects to be retrieved and processed so that a series of approximate answers of improving accuracy are produced.

4. Monotone Query Processing

APPROXIMATE uses a monotone query processing algorithm to produce an approximate answer and maintains semantic information for an effective implementation of this algorithm. As an alternative to processing the possible tuples during query processing, APPROXIMATE works on templates of the possible tuples P in an approximate relation. The approach used in APPROXIMATE to generate templates of possible tuples is similar to the one used to produce intensional answers [7]. APPROXIMATE maintains an object-oriented view of the database and uses the information provided by this view to generate the templates. In this view, a base relation, or a segment of the relation, is a class. Tuples in the relation, or the segment, are instances of the corresponding class. The classes are organized into a collection of class hierarchies. Each class hierarchy supplies information about a base relation stored in the database. Examples of the types of information provided by a class hierarchy include the domains of the attributes of the instances of a class and the retrievable unit of data. This

information is accessed along with the base relations during query processing.

The basic monotone query processing algorithm works as follows. It begins by representing a query by a query tree. Each node in the query tree represents a relation that is the result of a relational operation. An initial approximation is assigned to every node in the query tree. This initial approximation, and subsequent improved approximations of the standard relation represented by the node, are stored in an approximate object, which is created by APPROXIMATE for the node before query processing starts. The value of this approximate object gives an approximate relation of the standard relation. The object has three variables: the certain_part, possible_part, and OP.

The value of its certain_part is a certain set C containing all the data objects that are certainly in the standard relation. Initially, this certain set is empty for every approximate object. The value of its possible_part is a set **P** of possible classes. The set of all instances of the classes in **P** is a possible set in an approximation $(C, \mathbf{P})$ of the standard relation. Initially, the value of the possible_part in an approximate object at a leaf node gives a template of all possible tuples that can possibly be in the base relation represented by the leaf node. Again, this template is obtained from the meta data about the base relation maintained by the query processor. The initial values of the possible_part in an approximate object at a non-leaf node can be obtained from the initial values of the leaf nodes in the the subtree rooted at the node and the relational algebra operations represented by the nodes in the subtree. The value of the variable OP is set to be the relational algebra operation represented by the node. The OP of the approximate object at a leaf node is an approximate_read. An approximate_read returns a segment of the requested base relation at a time. Again, information on which segments can be returned is given by the view maintained by the query processor.

Figure 2 shows the value $\mathbf{A}_2$ of an approximate object corresponding to the relation A_2 in Figure 1(c). The possible tuples in P_2 are instances of $\mathbf{P}_2 = \{\text{ORD-long-runways}\}$, the possible class of long runways at ORD airport.

Because standard relational algebra operations cannot operate on approximate objects, APPROXIMATE uses as query processing primitives a set of approximate relational algebra operations and the approximate_read. Each operation accepts an approximate object(s) as an operand and produces an approximate object as its result [3]. As each approximate_read of a leaf node is carried out, each returned segment causes additional certain tuples to be added to, and possible classes to be deleted from, the current approximation of the base relation. The value of the leaf node improves as more segments are returned. The improvement in the leaf nodes is propagated upward to the root node by reevaluating the nodes in the query tree. The value of the root node is updated with better

id	Airport	Length
6	MDW	7500
7	MDW	8000
10	ORD	8000
{ORD-long-runways}		

Figure 2. $\mathbf{A}_2$: An approximate object

values each time the root node is reevaluated.

This monotone query processing algorithm differs from traditional query processing where each node of the query tree is evaluated only when all of the required data are available. If any required base relation is not accessible, no answer is ever produced. This all-or-nothing query processing strategy does not degrade gracefully. In contrast, APPROXIMATE produces a chain of increasingly better approximate answers at the root node of the query tree, each integrating the effect of additional data processed. None but the final, exact answer requires all base relation data be available before it can be produced. If query processing terminates prematurely, some approximate answer in the chain will be returned and the quality of the returned answer increases monotonically with time.

Every approximate relational algebra operation involves operations on the possible classes as well as the certain tuples in its operand(s). The approximate relational algebra operations are implemented incrementally, so the same number of relational algebra operations is applied to the certain tuples during approximate query processing as during traditional query processing [9]. To minimize the overhead required to produce an approximate answer, APPROXIMATE delays the evaluation of the possible classes and instead maintains a symbolic expression of the possible classes and relational algebra operations to be applied to them. A possible class is not evaluated until query processing must terminate and an approximate answer is produced or the user requests the evaluation.

With each update to the value of the root node of the query tree, APPROXIMATE displays the certain tuples and the possible class names of the approximate answer. Figure 3 illustrates two approximate answers to the query "Select all the northerly and easterly runways at airports where the temp > 32° C". As this figure illustrates, an approximate answer allows the user to distinguish the data processed thus far from the data not yet processed. From the approximate answer in Figure 3(a), the user can see that the classes {IL-short-N} runways and {IL-short-E} runways have not been processed yet. Upon examining the approximate answer, a user may determine whether the approximate answer resulting from processing all of the long runways provides enough information and is a good enough answer. The user can request the evaluation of the possible classes, and Figure 3(b) illustrates the evaluation of the possible classes in Figure 3(a). Figure 3(c) illustrates an improved approximate answer.

5. Future Directions

We have described a monotone query processor called APPROXIMATE that produces an approximate answer if some of the data are not available or if there is not enough time to process a query. APPROXIMATE processes the data that are available or processes the data that can be processed within the time constraints. It produces a series of approximate answers that improves according to a partial-order relation. The same query processing strategy is used for producing approximate answers to set-valued queries and for single-valued queries, such as binary queries, aggregate queries, and set-valued queries with cardinality 1. APPROXIMATE uses semantic support, in the form of an object-oriented view and a predefined set of distance functions, to identify a good initial approximation and a strategy for improving an approximation.

In the future, we will determine the scalability and efficiency of the proposed scheme. To accomplish this, we need to interface the query processor to some real-life database systems and make the query processor as efficient and robust as we can. Some candidate databases are those used to support navigation, machine vision, and computer aided engineering, as well as databases on students'

id	Airport	Length	Temp
1	ORD	11000	35
2	CMI	10000	38
7	ORD	10500	35
33	MDH	11000	50

Possible Classes:
{IL-short-N, IL-short-E}

Figure 3(a). An approximate answer

id: { 32-36, 1-13 }
Airport: { CMI,JOT,MDH,MDW,ORD,SPI }
Length: { 7000-9500 }
Temp: { > 32 }

Figure 3(b). Possible class evaluation

id	Airport	Length	Temp
1	ORD	11000	35
2	CMI	10000	38
7	ORD	10500	35
33	MDH	11000	50
36	MDW	8000	35

Possible Classes: {IL-short-E}

Figure 3(c). An approximate answer

academic reports. A natural extension of this research is to consider the case when the database contains incomplete or partial values. This problem is closely related to the imprecise update problem. Imprecise updates introduce incomplete information and partial values into the database. We want to investigate the feasibility of partial updates that require no error recovery action to complete the update. Such an update may introduce uncertainty or incompleteness in the information contained in the database, but will not lead the database to an unsafe, and hence, unacceptable state.

References

[1] Lin, K. J., S. Natarajan, J. W. S. Liu, and T. Krauskopf, "Concord: A System of Imprecise Computations," *Proc. COMPSAC '87*, Tokyo, Japan, pp. 75-81, Oct. 1987.

[2] Chung, J. Y., J. W. S. Liu, and K. J. Lin, "Scheduling Periodic Jobs that Allow Imprecise Results," *IEEE Transactions on Computers*, Vol. 39, No. 9, pp. 1156-1174, Sept. 1990.

[3] Vrbsky, S. V. and J. W. S. Liu, "An Object-Oriented Query Processor That Produces Monotonically Improving Approximate Answers," *7th International Conference on Data Engineering*, Japan, pp. 472-481, April 1991.

[4] Buneman, P., S. B. Davidson, and A. Watters, "A Semantics for Complex Objects and Approximate Queries," *Proceedings of the 7th Symposium on the Principles of Database Systems*, pp. 305-314, March 1988.

[5] Ozsoyoglu, Bultekin, Z. Meral Ozsoyoglu and Wen-Chi Hou, "Research in Time- and Error-Constrained Database Query Processing," *Workshop on Real-Time Operating Systems and Software*, Virginia, May 1990.

[6] Motro, Amihai, "VAGUE: A User Interface to Relational Databases that Permits Vague Queries," *ACM Transactions on Office Information Systems*, Vol. 6, No. 3, pp. 187-214, July 1988.

[7] Chu, Wesley W., Rei-Chi Lee and Qiming Chen, "Using Type Inference and Induced Rules to Provide Intensional Answers," *7th International Conference on Data Engineering*, Japan, pp. 396-403, April 1991.

[8] Smith, Kenneth P., and J. W. S. Liu, "Monotonically Improving Approximate Answers to Relational Algebra Queries," *Proc. COMPSAC '89*, Orlando, Florida, Sept. 1989.

[9] Vrbsky, S. V. and J. W. S. Liu, "APPROXIMATE: A Query Processor That Produces Monotonically Improving Approximate Answers," submitted to *IEEE Trans. on Knowledge and Data Engineering*.

A PRELIMINARY EMPIRICAL EVALUATION OF VIRTUAL REALITY AS AN INSTRUCTIONAL MEDIUM FOR VISUAL-SPATIAL TASKS

J. Wesley Regian[1], Wayne Shebilske[1], and John M. Monk[2]
[1]U.S.A.F. Armstrong Lab
AL/HRTI
Brooks AFB, TX 78235
[2]Galaxy Scientific Corp.
AL/HRMT Bldg. 6321
Lackland AFB, TX 78236

We explored the training potential of Virtual Reality (VR) technology. Thirty-one adults were trained and tested on spatial skills in a VR. They learned a sequence of button and knob responses on a VR console and performed flawlessly on the same console. Half were trained with a rote strategy; the rest used a meaningful strategy. Response times were equivalent for both groups and decreased significantly over five test trials indicating that learning continued on VR tests. The same subjects practiced navigating through a VR building, which had three floors with four rooms on each floor. The dependent measure was the number of rooms traversed on routes that differed from training routes. Many subjects completed tests in the fewest rooms possible. All subjects learned configurational knowledge according to the criterion of taking paths that were significantly shorter than those predicted by a random walk as determined by a Monte Carlo analysis. The results were discussed as a departure point for empirically testing the training potential of VR technology.

A Decision-Theoretic Approach to the Display of Information for Time-Critical Decisions: The Vista Project

Eric Horvitz* **Corinne Ruokangas** **Sampath Srinivas**
Palo Alto Laboratory
Rockwell International Science Center
444 High Street
Palo Alto, California 94301
and
Matthew Barry
Propulsion Systems Section
NASA Johnson Space Center
Houston, Texas

Abstract

We describe a collaborative research and development effort between the Palo Alto Laboratory of the Rockwell Science Center, Rockwell Space Operations Company, and the Propulsion Systems Section of NASA Johnson Space Center to design computational tools that can manage the complexity of information displayed to human operators in high-stakes, time-critical decision contexts. We shall review an application from NASA Mission Control and describe how we integrated a probabilistic diagnostic model, and a time-dependent utility model, with techniques for managing the complexity of computer displays. The, we shall describe the behavior of VPROP, a system constructed to demonstrate promising display-management techniques. Finally, we shall describe our current research directions on the Vista II follow-on project.

1. Introduction

The Vista project was established to develop computational techniques for reducing the cognitive load of human operators that are responsible for monitoring complex systems. Decision makers under pressure to take swift action cannot afford to sift through large quantities of potentially relevant information before making a decision. Fundamental limitations in the abilities of people to process information explain why the quality of a system operator's decisions can degrade as the complexity of relevant data increases, and as the time available for a response decreases. Cognitive psychologists have found that humans cannot retain more than five to nine distinct concepts or "chunks" of information simultaneously (Miller 1956). This surprising cognitive limitation was demonstrated in a classic study by Miller, and has been confirmed repeatedly by experimental psychologists. The capacity of decision makers to consider important influences on a decision may be reduced even further if fast action is demanded in frenetic crisis situations; one cognitive-psychology study showed that people cannot retain and reason simultaneously about more than two concepts in environments filled with distractions (Waugh and Norman 1965).

There has been previous investigation of methods for managing the complexity of computational results to decrease the cognitive burden of computer users. Key concepts employed in the Vista project are similar to techniques developed in related work on the

*Also at Knowledge Systems Laboratory, MSOB X215, Departments of Computer Science and Medicine, Stanford University, Stanford, CA 94301

control of computation and decision making under bounded resources (Horvitz, et al. 1989, Horvitz 1990, Horvitz and Rutledge 1991). In the related work on the decision-theoretic control of computation, investigators have examined decisions about the tradeoff between precision and timeliness of generating a computational result (such as a recommendation for action in the world). The analogous techniques for controlling displays have centered on decisions about the tradeoff between the completeness and the complexity of information displays and computer-based explanations (Horvitz, et al. 1986; Horvitz, et al. 1989). In some of this work, multiattribute utility has been employed to control the complexity of displays (Horvitz 1987; Mclaughlin 1987) and for queuing and prioritizing the results of diagnostic reasoning (Breese, et al. 1991).

We have worked to integrate a set of flexible display strategies with a probabilistic reasoning module that performs diagnosis under uncertainty of propulsion systems of the Space Shuttle. The diagnostic-reasoning component of the approach continues to reason about the likelihood of alternative disorders in a system based on a continuous stream of sensor information. The Vista team consists of a close collaboration between members of the Palo Alto Laboratory group (Eric Horvitz (PI), Corinne Ruokangas, and Sampath Srinivas) and a propulsion systems specialist at Houston Mission Control (Mathew Barry). In our approach, we employ model-based diagnostic reasoning and flexible displays for reconfiguring the information displayed so that the most relevant information is presented in different context.

We tested our ideas about the model-based control of displays by building and validating a display manager named VPROP. The techniques explored in VPROP have application in many domains where decision makers with limited time must evaluate large amounts of information under time pressure to identify critical data.

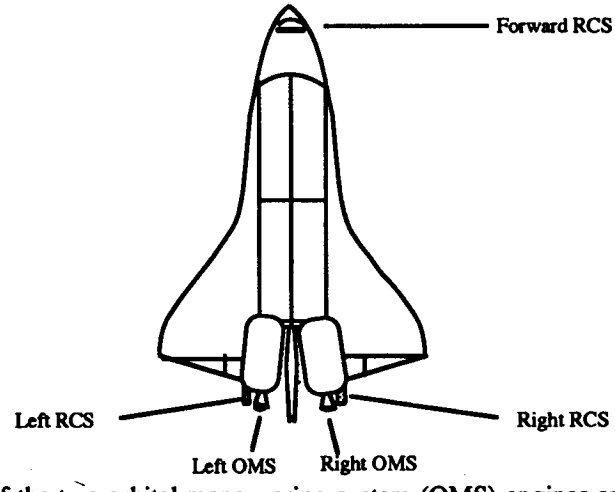


Figure 1. The position of the two orbital maneuvering system (OMS) engines and the three sets of reaction control system (RCS) thrusters on the Space Shuttle.

2. Application Area

We examined problems with the complexity of information displays used by ground controllers at Space Shuttle Mission Control in Houston. In particular, we worked with experienced flight controllers who manage Space Shuttle propulsion systems. The Propulsion Team is responsible for monitoring two different space-based thruster systems: the Orbital Maneuvering System (OMS) and the Reaction Control System (RCS). The large right and left OMS engines are fired for such critical maneuvers as orbital insertion and orbit circularization. The smaller suites of RCS thrusters are used for translation in space, for such tasks as maneuvering near another space object, as well as for the continual

The status quo computer displays for monitoring the propulsion systems consist of a cluttered, static main display of the two OMS engines, and the three banks of RCS engines. This primary propulsion-systems display is pictured in Figure 2. When ground controllers are concerned about one of the systems, alternate screens are typically requested which relay trend information about engine consumables by displaying a large table of numbers. An example of such a secondary display is pictured in Figure 3. The large amount and complex display of information that must be scanned in crisis situations can become burdensome in situations that demand quick decision making, especially for new people on the propulsion team.

3. Construction of a Probabilistic Model

The first task in developing a model-based display manager was the construction of a probabilistic diagnostic model for the shuttle propulsion-system engines. To build the diagnostic reasoner of the display manager, our team worked with propulsion experts at Mission Control to build a causal model of the shuttle's propulsion subsystems. Figure 4 displays a basic schematic of the OMS engine. In the engine, a tank of helium puts pressure on tanks of oxidizer and fuel. To fire an OMS engine, a bipropellant valve is opened which allows the fuel and oxidizer to mix and combust to provide thrust. In addition to the basic flows, ground controllers must also consider the status of a set of values between various tanks, and crossover lines that allow fuel to be shared by different engine systems. There are suites of temperature and pressure sensors at critical locations in the system. Shuttle telemetry includes information from these sensors.

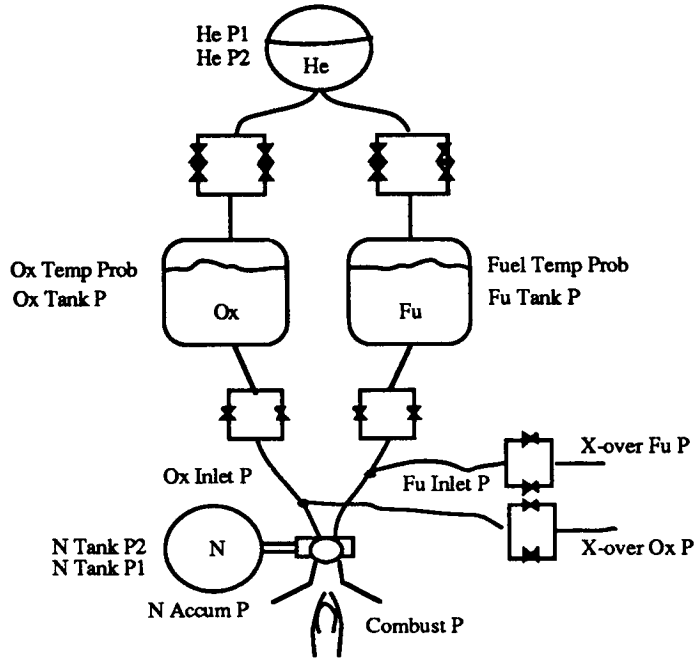


Figure 4. A schematic of an OMS engine.

We constructed a probabilistic causal network, called a belief network, to model the uncertain relationships among components of the system. Belief networks serve as the core of an increasing number of probabilistic reasoning systems. (See (Pearl, 1989) and (Horvitz, 1988) for reviews of probabilistic and decision-theoretic reasoning.) In practice, a domain expert, working with a computer engineer, structures and assesses the probabilistic relationships in a

belief network. In this case, Matthew Barry, a propulsion specialist at NASA Mission Control served as expert. The belief network representation allows an expert to structure relationships about a system qualitatively, and, after, to quantify those relationships with conditional probabilities.

Belief networks serve as the kernel of diagnostic reasoning systems. Technically, a belief network is a directed acyclic graph (DAG) containing nodes, representing propositions (hypotheses, intermediate states, and observations), and arcs representing probabilistic dependencies among nodes. Nodes are associated with a set of mutually exclusive and exhaustive values that represent alternative possible states of a proposition (*e.g.*, true, false). Directed arcs capture knowledge that the value of a parent node can affect the probability distribution over the values of children nodes.

A variety of belief-network algorithms have been formulated. The algorithms propagate information through the network to compute the probability distributions over values of each node, given the observation of one or more values of a subset of nodes that represent sensors or observations. The belief network allows us to represent the notion that there may be multiple causes of problems, and the related notion that a sensor may fail, and thus provide erroneous information about the system being monitored.

Figure 5 represents an early version of a belief network constructed to model a Shuttle OMS engine. Following arcs through a belief network often tells us a story about uncertain relationships in a system. For example, the value of HELIUM PRESSURE affects the pressure readings (He P1, He P2) reported by the two independent pressure sensors on an OMS helium tank. However, the readings can also be affected, with uncertainty, by the errors in the sensor mechanisms themselves. A user with experience in sensor failures can encode his belief about the relative rate of failure of alternative critical sensors in a system.

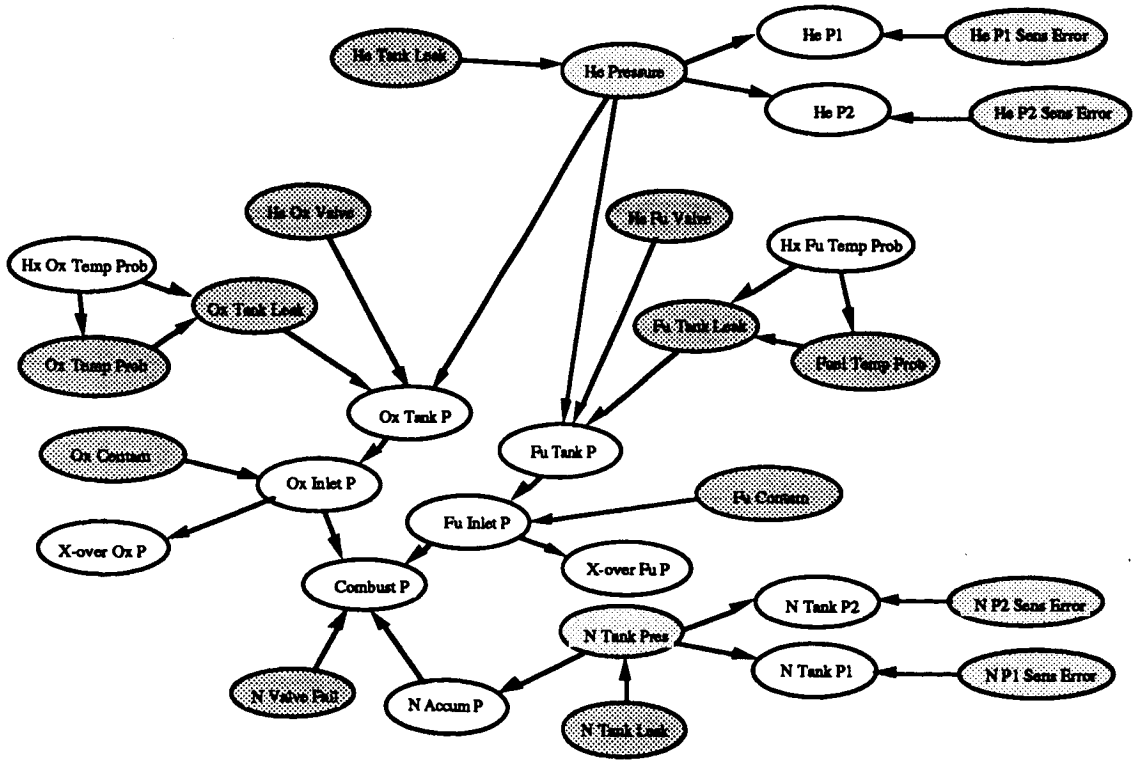


Figure 5. A causal probabilistic network for the OMS engine.

4. Display Management in VPROP

The diagnostic reasoning system was integrated with a set of display complexity-management techniques to yield a dynamic display system. Key features of VPROP's display include (1) the use of context-sensitive templates for configuring data for different phases of the mission, (2) a static screen layout of panels of information on key systems, (3) the problem-specific telescoping of system information along a simplicity--complexity vector, (4) the summarization of large quantities of information with simple status indicators, (5) the manual access of any data through an iconic information palette, and (6) the use of probabilistic and decision-theoretic reasoning to dynamically prioritize menus for accessing critical information for decision making.

4.1 Techniques for Managing Display Complexity

Figure 6 highlights these flexible display techniques. In this case, panels of information are configured for a critical OMS firing. For this context, system-specific panels containing information on the left, forward, and right RCS systems are reduced into brief summaries and information panels on the left and right OMS are expanded. By moving the cursor and clicking on any system-specific panel, a menu can be accessed that allows the detail, and concomitant area occupied by information about a particular system, to be increased or decreased beyond the standard configuration for a context.

We can simplify information about a Space Shuttle system as we shrink a panel by modulating the abstraction or the completeness of information (Horvitz, 1987). For VPROP, the expert defined dimensions of complexity which described how the details about a propulsion system should change, as a panel is allowed to occupy more screen area. That is, an expert decision maker defined how the content and granularity of information in each panel changes as that panel is enlarged or diminished.

A *display palette* appears at the lower left hand corner of the display. The display palette is laid out in a configuration that iconically reflects the main screen layout of the system-specific information panels. By clicking on soft buttons in the palette for each system, the main panels of information are incremented in detail and then reduced, as if alternate levels of complexity were stored in a circular queue. The palette doubles as a systems-status overview and summary. If telemetry about a system is within normal bounds, the corresponding display-palette soft button appears green. If there is a problem with a system, the corresponding display-palette button flashes red.

The usefulness of employing abstraction to simplify the system-specific panels and to alert an operator about system status is supported in part by findings of cognitive psychologists that people employ abstraction of items into classes to manage the complexity of reasoning about complex problems (Mesarovic 1970, Simon 1973).

4.2 Integrating Display-Complexity Tools with Diagnostic Reasoning

A belief network, that we constructed and assessed with the assistance of an experienced ground controller, is employed in VPROP to continue to monitor Space Shuttle telemetry on propulsion systems. Figure 6 highlights VPROP's reaction when a problem with the left OMS has been noted. In this case, the probability that the left OMS is normal has dropped below a threshold value. The left-OMS button in the display palette turns red and the system-specific information panel for the the left OMS automatically expands in size and complexity.

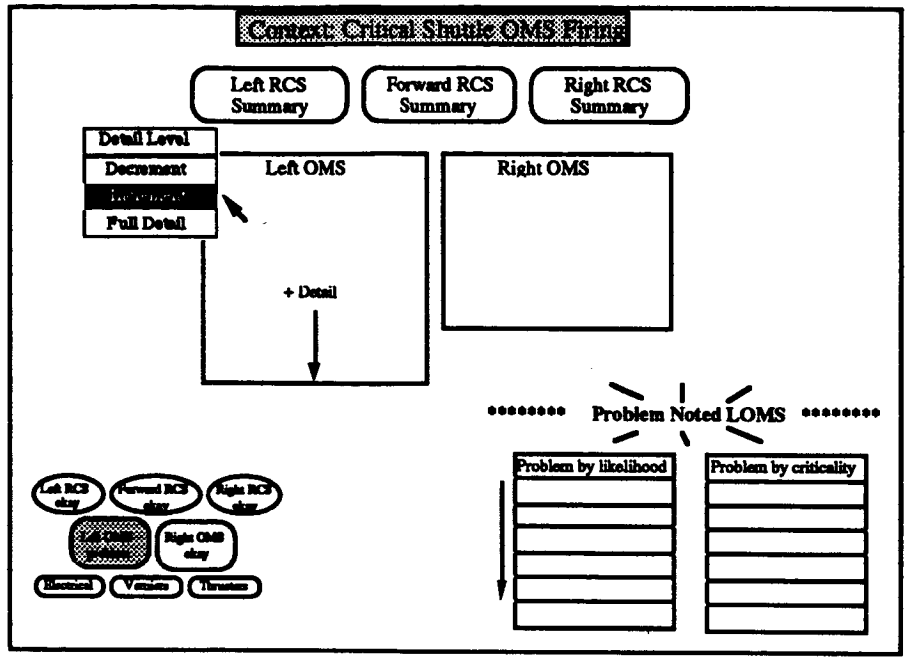


Figure 6. Key components of VPROP's flexible display.

Now, the diagnostics on the left-OMS are displayed, as portrayed by the two lists, displayed at the lower right-hand-corner of Figure 6. A summary of the problem is stated and a list of possible disorders computed by the belief network are listed by likelihood. In another column, the disorders are reordered into a set of priorities, in terms of the *expected time-criticality* of the problems. The expected time-criticality is computed by weighting a relative cost of delay for each possible disorder, assessed from the ground controller, by the likelihood computed for that disorder.

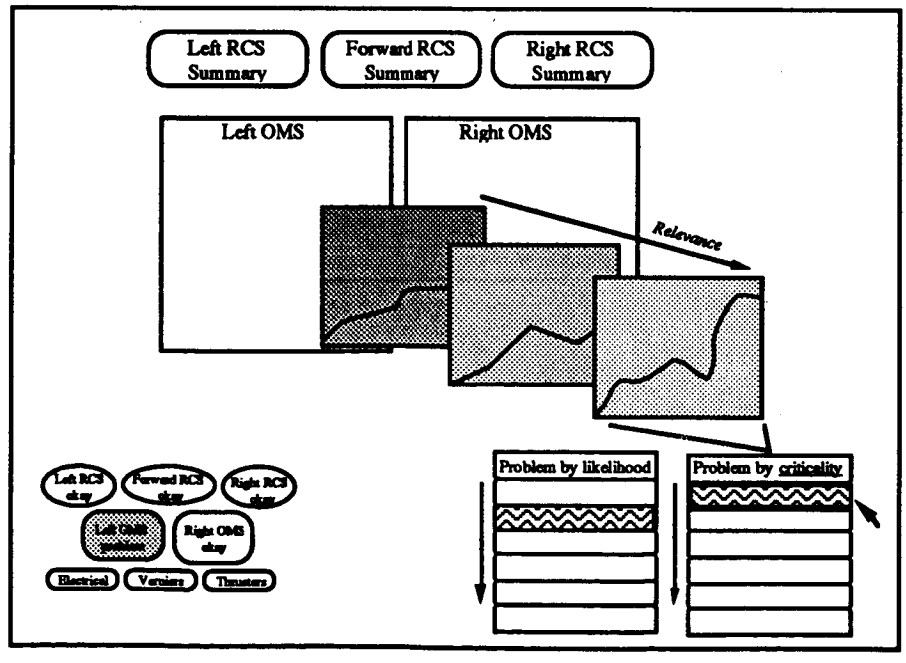


Figure 7. Accessing trend information from a prioritized list of possible faults.

Information about trends is often used by Shuttle-propulsion ground controllers to confirm disorders with the propulsion systems. VPROP allows ground controllers to access trend graphs, by positioning a cursor and clicking a mouse button on disorders appearing on the likelihood or criticality lists. Figure 7 highlights how VPROP displays the trend information in a predefined order of relevance.

The integrated Vista methodology for managing the complexity of displays, while providing diagnostic information about the likelihood of alternative faults and recommendations about time-dependent priorities, is highlighted in Figure 8. As indicated in this figure, a goal of the Vista project is to field systems that analyze real-time shuttle telemetry with probablistic causal models and that consider the most valuable configuration of the display in terms of the likelihood and time-dependent losses associated with different faults.

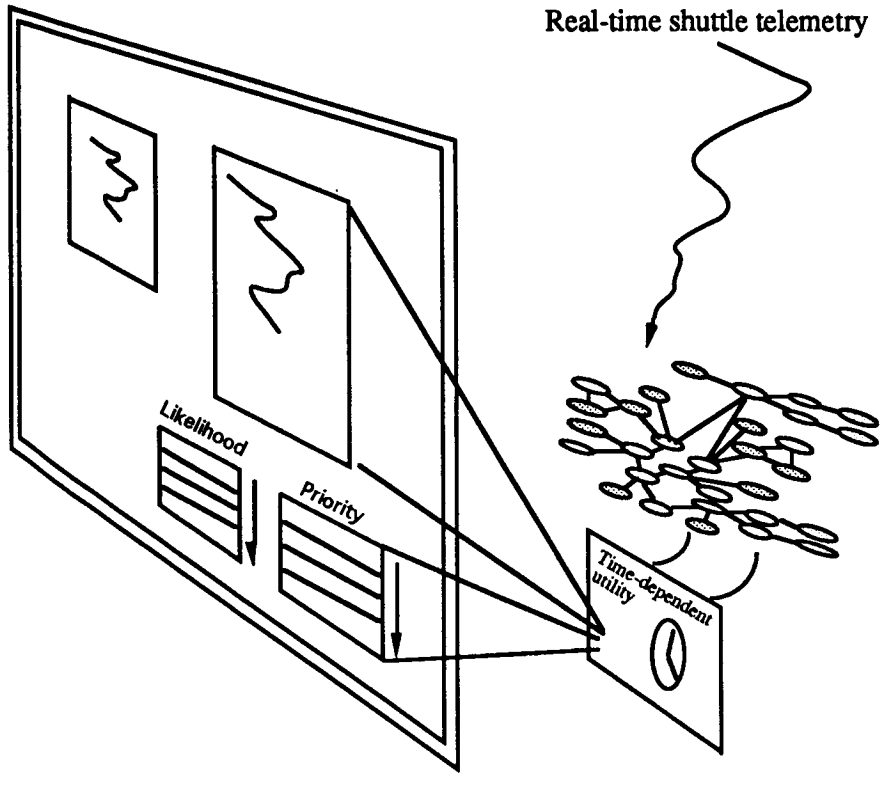


Figure 8. Key components of the Vista approach. A belief network is used to consider the diagnostic relevance of shuttle telemetry. A model of time-dependent utility is employed to prioritize alternative possible faults, given the likelihood of the faults, and to control the size and amount of detail dedicated to a panel describing the system in which the possible faults may be occuring.

A bitmap of the current VPROP screen is displayed in Figure 9. In this case, troubling telemetry has been detected by the belief network. A diagnostic screen appears, showing that, given the current sensor information, there is a high probability of engine failure. Relevant trend information has been accessed.

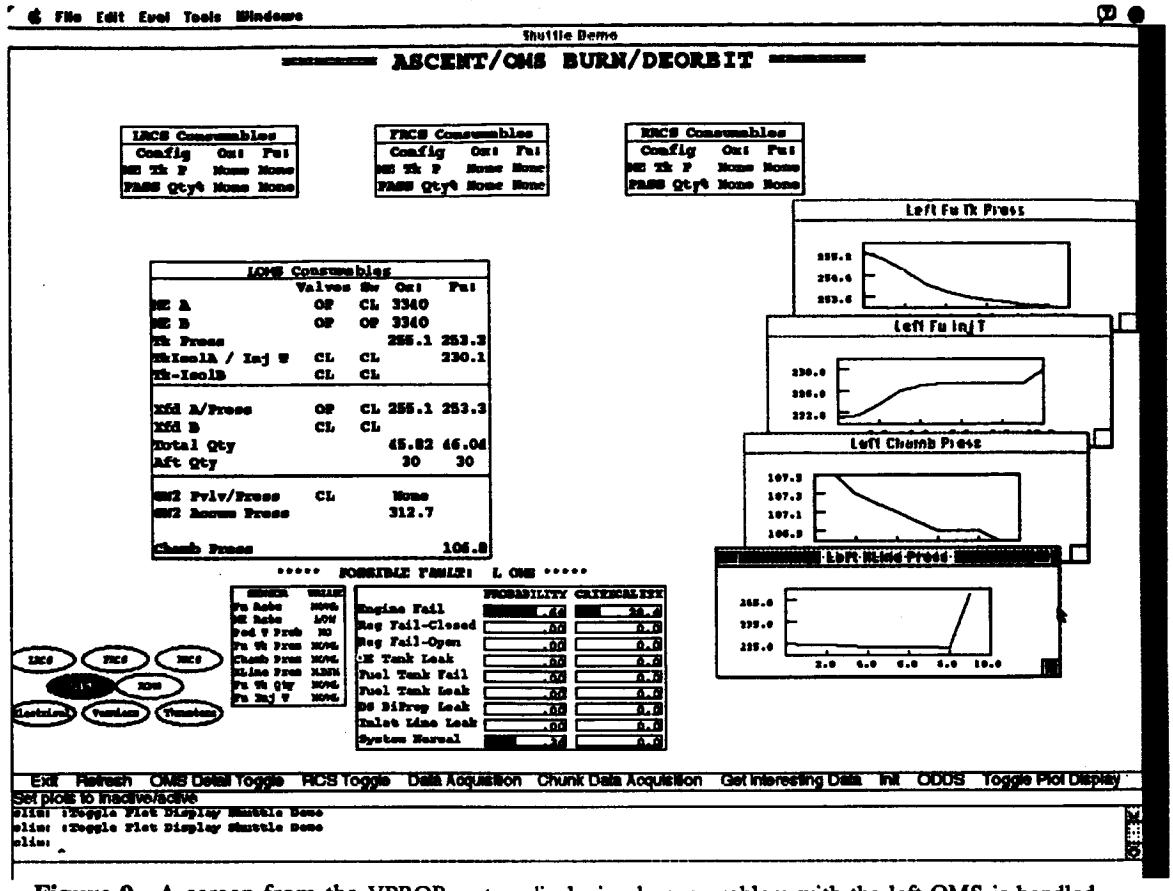


Figure 9. A screen from the VPROP system displaying how a problem with the left OMS is handled. In this case, there is a high probability of an engine failure. Relevant trend information is accessed.

5. Conclusions and Summary

The VPROP system has been reviewed by the Propulsion Team at NASA Mission Control in Houston. The system reviews were met with enthusiasm about the approach and future extensions to the methodology. Discussions and research among the Palo Alto Laboratory, the Rockwell Space Operations Company, and NASA Johnson Space Center are continuing on future projects to extend and apply Vista technology for monitoring and controlling other Space Shuttle systems as well as systems on the planned Space Station. The ability to consider alternative hypotheses under uncertainty promises to be even more crucial for monitoring systems on the space station; in comparison to the Shuttle, current plans call for the Space Station to have relatively few sensors on complex systems. This situation will make it more difficult to narrow a problem down to a single fault, given telemetry information.

Recently, we initiated the Vista II project, a Vista follow-on effort to develop a version of VPROP to be deployed on Unix workstations. The project centers on building an effective integrated display-management and diagnostic reasoning architecture, as well as the development of a new display management language for increasing the efficiency with which we can construct and custom-tailor reasoning systems for different ground-control ans space-based monitoring functions.

The integration of flexible display techniques and model-based systems with the ability to reason under uncertainty promises to provide a valuable framework for managing the cognitive load of human operators. We hope that the combination of intelligent diagnostic and display systems will allow us to keep pace with the growing complexity of machines and tasks. We

foresee that intelligent reasoning systems will be important in helping to maintain--or even to reduce--the cognitive burden on human operators charged with monitoring or managing systems of increasing complexity. Indeed, the design of computational tools for managing the complexity of information about complex systems may one day be viewed as an intrinsic component of the design of the systems.

Acknowledgments

We thank John Breese, Max Henrion, Jim Martin, Michael Shaff, Philip Stubblefield, and Cynthia Wells for providing valuable feedback and assistance on Project Vista.

References

(Breese, et. al 1989)
J.S. Breese, S. Srinivas, and R.A. Fiske. A Utility-Based Pilot Vehicle Interface. *Proceedings of DARPA Symposium on Associate Technology,* pages 259-266. George Mason University, Fairfax, VA, June, 1991.

(Horvitz 1987)
E.J. Horvitz. *A Multiattribute Utility Approach to Inference Understandability and Explanation.* Technical Report KSL-87-28, Knowledge Systems Laboratory, Departments of Computer Science and Medicine, Stanford University. March, 1987.

(Horvitz 1990)
E.J. Horvitz. *Computation and Action Under Bounded Resources.* Ph.D. Dissertation, Departments of Computer Science and Medicine, Stanford University, December, 1990.

(Horvitz, et al. 1986)
E.J. Horvitz, D.E. Heckerman, B.N. Nathwani, and L.M. Fagan. The use of a heuristic problem-solving hierarchy to facilitate the explanation of hypothesis-directed reasoning. In *Proceedings of Medinfo,* Washington, DC, pages 27-31. North-Holland, New York, October 1986.

(Horvitz, et al. 1988)
E.J. Horvitz, J. Breese, and M. Henrion. Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning, Special Issue on Uncertain Reasoning in Artificial Intelligence.* 2:247-302.

(Horvitz, et al. 1989)
E.J. Horvitz, D.E. Heckerman, K. Ng, and B.N. Nathwani. Heuristic abstraction in the decision-theoretic Pathfinder system. In *Proceedings of the Thirteenth Symposium on Computer Applications in Medical Care,* Washington, DC, pages 178-182. IEEE Computer Society Press, Silver Spring, MD, 1989.

(Horvitz, et al. 1989)
E.J. Horvitz, G.F. Cooper, and D.E. Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, MI, pages 1121-1127. Morgan Kaufmann, San Mateo, CA, August 1989.

(Horvitz and Rutledge 1991)
E.J. Horvitz and G. Rutledge. Time-dependent utility and action under uncertainty. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence,* Los Angeles, CA, pages 151-158. Morgan Kaufmann, San Mateo, CA, July 1991.

(Mclaughlin 1987)
J. Mclaughlin. *The Utility-Directed Presentation of Graphical Simulation*. Technical Report KSL-87-59, Knowledge Systems Laboratory, Departments of Computer Science and Medicine, Stanford University. December, 1987.

(Mesarovic 1970)
M.D. Mesarovic, D. Macko, and Y. Takahara. *Theory of Hierarchical, Multilevel Systems*. New York: Academic Press, 1970.

(Miller 1956)
G.A. Miller. *The Magical Number Seven, Plus or Minus Two*. Psychological Review, 63: 81-97.

(Simon 1973)
H.A. Simon. *The Organization of Complex Systems* in Hierarchy Theory, H.H. Pattee, ed., pages 3-27. G.Brazilier, 1973

(Waugh and Norman 1965)
N.C. Waugh and D.A. Norman. *Primary Memory*, Psychological Review. 72:89-104.

COOPERATIVE ANSWERS IN DATABASE SYSTEMS *

Terry Gaasterland, Parke Godfrey, Jack Minker, and Lev Novik
Department of Computer Science
A.V. Williams Building
University of Maryland
College Park, MD 20742

1 Introduction

A major concern of researchers who seek to improve human-computer communication involves how to move beyond literal interpretations of queries to a level of responsiveness that takes the user's misconceptions, expectations, desires, and interests into consideration. At Maryland, we are investigating how to better meet a user's needs within the framework of the cooperative answering system of Gal and Minker [25]. We have been exploring how to use semantic information about the database to formulate coherent and informative answers. The work has two main thrusts: 1) the construction of a logic formula which embodies the content of a cooperative answer; and 2) the presentation of the logic formula to the user in a natural language form. The information that is available in a deductive database system for building cooperative answers includes integrity constraints, user constraints, the search tree for answers to the query, and false presuppositions that are present in the query. The basic cooperative answering theory of Gal and Minker [15, 25] forms the foundation of a cooperative answering system that integrates the new construction and presentation methods.

This paper provides an overview of the cooperative answering strategies used in the CARMIN cooperative answering system, an ongoing research effort at Maryland. Section 2 gives some useful background definitions. Section 3 describes techniques for collecting cooperative logical formulae. Section 4 discusses which natural language generation techniques are useful for presenting the logic formula in natural language text. Section 5 presents a diagram of the system.

2 Some Definitions

Deductive databases are comprised of syntactic information and semantic information. The syntactic information consists of the intensional database (IDB) which is the set of clauses of the form $A \leftarrow B_1, \ldots, B_n$, $n > 0$, where A and each B_i is an atom, and the extensional database (EDB) which is the set of clauses of the form $A \leftarrow$. IDB clauses are also called rules. EDB clauses are also called facts.

*This paper describes research done at the Computer Science Department at the University of Maryland under Air Force Office of Scientific Research grant AFOSR-91-0350 and NSF grant IRI-89-16059.

Direct answers to database queries, which are clauses of the form $\leftarrow B_1, \ldots, B_n$ are found by using SLD-resolution on the query, IDB, and EDB clause to produce a search tree. The root node of the search tree is the query clause; each node in the tree is produced by applying an IDB rule to the node above. A successful leaf node is one which matches with EDB facts; a failed leaf node is one which has no match in the EDB [16, 20].

The semantic information about the database consists of a set of integrity constraints (IC). Semantic information about users of the database consists of a set of user constraints (UC). The constraints considered in this paper have the form $\leftarrow C_1, \ldots, C_n, E_1, \ldots, E_m$, where each C_i is an atom whose predicate appears in an EDB fact or the head of an IDB rule and each E_i is an evaluable expression.

An integrity constraint restricts the states that a database can take. For example, the integrity constraint, *"No person can be both male and female,"* $\leftarrow$ *person(X),male(X),female(X)*, restricts people in a database from having both properties. Because the constraints on a database do not enable the deduction of new answers but rather give information about existing knowledge and answers, they are considered semantic information rather than syntactic information.

A user constraint reflects restrictions that the user places on the database. For example, a user who detests Kennedy airport may impose a restriction on a database about traveling that he does not want to travel into JFK at any point in a trip: $\leftarrow$ *travel(X,JFK)*. In practice, user constraints are labeled as applicable to some particular user.

3 Constructing the Logic Formula

The cooperative answering system of Gal and Minker [25], the precursor to the CARMIN system, uses two basic strategies to identify extra information which can be included in an answer: 1) it checks whether a query is restricted by or violates any of the database's integrity constraints; and, 2) for failed queries, it searches for any false presuppositions in the query. Integrity constraints can help identify a user's misconceptions about the database that are made evident by the user's query. When an integrity constraint interacts with a query and shows that part of its search space must fail, the interaction indicates that the user does not know about some of the information embodied in the constraint. A description of the constraint to the user can be used to correct such misconceptions.

Some queries can have successful answers only if the database is in a state which conflicts with some integrity constraint. Since the database will never take such a state, the query will never succeed. The failure of such a query can be explained through an integrity constraint with which it conflicts. Such a constraint represents a misconception that the user had about the database. For example, if a query asks for someone who is both a mother and a father, that is, *"who is a parent and female and a parent and male?"*, it conflicts with the constraint that no one can be both male and female. Integrity constraints are also useful for identifying queries that contain redundancies, like *"who are all the males who are not female?"*. A response which includes paraphrases of the involved constraints can help correct a user's misconceptions about the database.

The presuppositions in a query are statements that must be true for either the query or its negation to be true [19]. For example, *"list the mothers in the database."* presupposes that mothers exist in the database. A query with a false presupposition has no answer, positive or negative, and a cooperative response would identify the false presupposition to the user.

3.1 Using User Constraints

In an extension to the cooperative answering system, user constraints are used to take into account a user's restrictions on the world when answering questions. A user's intentions and needs can be modeled as a set of constraints. User constraints reflect the semantics that a particular user imposes on the database. They are provided by individual users and need not be consistent with the database. When answering queries posed by a user, they are used to modify a user's query so that the search space of the original query is limited to find only answers that are amenable to the user's needs. The semantic compilation method of Chakravarthy et al. [3] which Gal and Minker use to incorporate integrity constraints into a query can also be applied to user constraints.

Consider a query to a database *"Which airline can I use to travel from Washington DC to Paris' Charles de Gaulle airport?"*:

Q: $\leftarrow$ *travel(Airline,Airport,CDG,Time)*$\wedge$ *near(Airport,Washington).*

and a user constraint *"I refuse to travel through JFK"*:

U: *Loc1*$\neq$ *JFK* $\leftarrow$ *flight(Airline,Number,Loc1,Loc2,Time).*

With database information that includes facts about flights from Washington National to JFK, JFK to CDG, and Washington Dulles to CDG, the query can be answer with *"Take PanAm to JFK and then to CDG."* However, such an answer is useless to the user - it violates the user constraint. An answer which satisfies the user's constraint provides the alternative which does not go through JFK, *"Take AirFrance from Dulles to CDG."*

When a set of answers has been restricted by a user constraint, it would be judicious to assume that the user knows about the user constraint – after all, the user supplied it. So, whereas selected integrity constraints are presented to the user whenever they apply to the query, user constraints are presented more sparingly.

3.2 The Search Tree

The search tree for the set of direct answers to a query contains abundant information both about the answers that are derived for the query and also about the failure paths. Each branch of the tree ends either in failure or success. Some of the failure branches may be labeled by integrity constraints; some may be labelled by user constraints; and some, by both. Some branches may be labelled by false presuppositions.

The cooperative answering system collects the intensional portion of the proof tree for a query as the IDB rules are applied to the query. This intensional proof tree, which we call the IDB-tree, includes all information necessary to connect the original query with any EDB level query literals which conflict with a constraint. It also includes extra information. We have developed an algorithm for gleaning from the IDB-tree the minimal information for making the connection. We then add the IDB-tree information to the cooperative response.

To illustrate the need for search tree information, consider an example about flights that serve meals. The following IDB rule defines the relation *meal_flight* for *breakfast*:

I: *meal_flight*(A,N,*breakfast*) $\leftarrow$

flight_times(A,N,T1,T2),
T1 <09:00, T2>10:00,
¬ *service_type*(A,N,*express*).

The rule says that a flight serves breakfast if the departure time is before 9:00am and the arrival time is after 10:00am and if the flight is not an express flight. Suppose the user poses a constraint that s/he does not want to take flights longer than one hour on United:

U: ← *flight_times*(A,N,T1,T2),A=*united*, Diff is T1 - T2, Diff > 01:00.

Suppose the user then asks a query about which flights serve breakfast between DC National and Chicago O'Hare:

Q: ← *meal_flight*(A,N,*breakfast*),*flight_airports*(A,N,*dcu*,*ohare*).

The original user constraint and the query have no common predicates, thus, the constraint does not affect the query directly. However, when the IDB rule is applied to derive the following subquery Q′, the constraint U partially subsumes Q′:

Q′: ← *flight_times*(A,N,T1,T2), T1 <09:00, T2>10:00,
flight_airports(A,N,*dcu*,*ohare*).

The residue, { ← A=*united* }, restricts the query variable A from taking the value *united*. Thus, all answers to the user will leave out flights on United Airlines.

Since the interaction between the constraint and the query occurred within the search tree, the user may not be aware of the interaction, and the user constraint should be included in the answer to the query. However, an answer that consists only of answer substitutions for the original query and the user constraint may be confusing to the user. The user must infer the connection between the constraint and the original query. For example, consider the following logic response, in which the components are labelled A for answer substitution and U for user constraint information:

A(*meal_flight*(*american*,101,*breakfast*) ∧ *flight_airports*(*american*,101,*dcu*,*ohare*)) ∧
U(← *flight_times*(A,N,T1,T2),
Diff is T1 - T2, Diff > 01:00,
A=*united*).

The response can be paraphrased as: *"American Airlines flight 101 flies from DC National to Chicago O'Hare and serves breakfast. You do not want to know about United flights that are longer than an hour."* The user must infer that a breakfast flight is longer than an hour. For a user who is not aware of this information, a better answer would include the following elaboration:

meal_flight(A,N,*breakfast*) ← *flight_times*(A,N,T1,T2), T1 <09:00, T2>10:00.

This may be paraphrased as *"a flight serves breakfast only if the flight starts before 9:00am and ends after 10:00am."* Notice that the elaboration does not contain the literal in the rule I about *ser-*

vice_type. The algorithm for selecting search tree information described in [10] adds the elaboration to the logical response.

3.3 Relaxation

As noted by many researchers, including [1, 4, 6, 18, 21, 27, 28, 30], an alternative form of cooperative behavior involves providing associated information which is relevant to a query. Generalizing a query in order to capture neighboring information is one means to obtain possibly relevant information.

Gaasterland, Godfrey and Minker have defined a method to *relax* a query in order to find neighboring information [12]. A query can be relaxed in at least three ways: 1) rewriting a predicate with a more general predicate; 2) rewriting a constant (term) with a more general constant (term); and 3) breaking a join dependency across literals in the query. The first two relaxations are achieved in a general manner using *taxonomy clauses* which define hierarchical type relationships between predicates and constants in the database language. For example, the following clauses define relationships between the predicates *communicate*, *call*, *mail*, and *say*:

T1: *communicate*(P_1, P_2, *Msg*) $\leftarrow$ *say*(P_1, P_2, *Msg*).
T2: *communicate*(P_1, P_2, *Msg*) $\leftarrow$ *lives_at*(*Addr*, P_2), *mail*(p_1, *Addr*, *Msg*).

A query containing a request to mail an invitation to the address *"34 Cherry Lane,"* upon failing, can be relaxed to produce alternative queries:

Q: $\leftarrow$ *mail(Terry, "34 Cherry Lane", invitation).*
Relaxed Q: $\leftarrow$ *communicate(Terry, P, invitation), lives_at("34 Cherry Lane", P).*

After the relaxation step, SLD resolution can be used to find related answers. [12] discusses search strategies for relaxation. The method has been incorporated into the cooperative answering system.

4 On Presenting the Cooperative Response in Natural Language

The cooperative answering strategies discussed in the previous section produce potentially large and complex logic formulae which are composed of logical expressions with a variety of origins, as seen in the breakfast flight example of Section 3.2. Even if the formula is presented in small portions, it still may be difficult for the user to read and understand the logic. The generation of natural language is a promising alternative method of presentation [2, 14].

Mapping each individual atom in a logical formula into an expression that emulates natural language is a straightforward process with the use of templates for each predicate. Unfortunately, the coherence and organization of the resulting text is limited to the organization that is accidentally present in the ordering of rules in the database and in the ordering of literals within the rules. In the direction of connecting individual pieces of text into a coherent whole, we have developed a series of linguistically motivated logic transformations that identify potential sites of coordination, subordination, and anaphora in the target text. We also have a method to select temporal

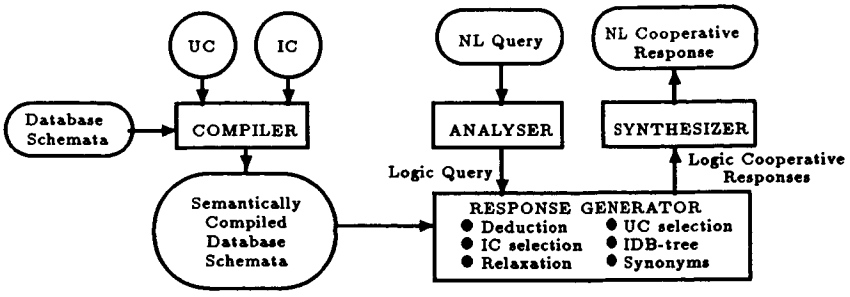


Figure 1: Cooperative Answering System At Maryland

connecting words, tense, and aspect [8]. To reduce redundancy across sentences, an additional transformation incorporates synonymous and generalizing substitutions into the target text.

The transformations take advantage of dependencies across literals, type information, and historical knowledge about how the formula was gathered. The coordination transformation uses a compact representation technique called π-notation [23] to help organize clauses. The computational theory of coordination used in the transformation builds on the work of [7, 9]. The selection of tense and aspect depends on temporal information available in the temporal database model of [31] and on temporal interval relationships [1]. This information enables the selection of allowable tenses using Hornstein's theory of tense [17], selection of aspectual feature values [5, 26] and selection of temporal connecting words such as "before," "while," "when," and "after." The incorporation of synonyms into the target text uses the taxonomy clauses and the relaxation method mentioned in the previous section to identify more concise representations of concepts in the logic formula.

Each transformation on the logic formula adds cohesion to or removes redundancy from the natural language text that is generated to describe the formula. The transformed logic formula is used as input to a natural language phrase generator to produce output text. The language generation work draws from and builds on work that has been done in the area of generating language from database information (e.g. [22]) and in the area of generating language from logic-based knowledge representations (e.g. [24, 32]). The language generation techniques used for cooperative answering with CARMIN are described in detail in [10].

5 Summary

The techniques discussed in this paper extend the cooperative answering system of Gal and Minker [25]. The user constraint, relaxation and search tree selection techniques complement the explanatory information provided by integrity constraints. They give information about how a user might construct new queries that will better serve the user's needs. CARMIN incorporates each of these techniques into a uniform user interface written in Prolog for relational and deductive databases. CARMIN provides a testbed for studying these and future cooperative techniques.

Figure 1 describes the structure of the cooperative answering system described here. Semantic query optimization techniques are used by a semantic compiler to integrate ICs and UCs with the database schema. For limited input language, a natural language analyzer borrowed from [29] parses natural language queries into queries in the language of the database — *logic queries*. The response generator processes a logic query with both the semantically compiled database schema

and the database itself to perform deduction, detection of IC and UC violations, restriction of the query with ICs and UCs, relaxation, and selection of information from the query's search tree. The response generator produces a cooperative response in the language of the database. The synthesizer then produces a natural language description of the logic response.

Semantic information in a deductive database is a relatively untapped source of information that can be used in responses to users' queries. Cooperative answers that use semantic information such as integrity constraints and user constraints can provide the user with information about the database's organization and the world modeled by the database. When queries fail or when a user wants additional answers, semantic information that describes taxonomy relationships between database predicates and constants can be used to find new queries that return answers related to the original query. For a background survey and comparison of approaches to cooperative answering, the reader is referred to [11].

References

[1] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–160, 1984.

[2] P. Amsili, A. Gal, F. Molines, P. Saint-Dizier, and E. Viegas. Some linguistic aspects of the generation of cooperative responses. In *ICO '91*, Montreal, Canada, 1991.

[3] U. Chakravarthy, J. Grant, and J. Minker. Foundations of semantic query optimization for deductive databases. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 243–274. Morgan-Kaufmann, 1987.

[4] W. W. Chu, Q. Chen, and R. Lee. Cooperative query answering via type abstraction hierarchy. Technical report, University of California at Los Angeles, October 1990.

[5] Bernard Comrie. *Aspect.* Cambridge University Press, Cambridge, England, 1976.

[6] F. Cuppens and R. Demolombe. How to Recognize Interesting Topics to Provide Cooperative Answering. *Information Systems*, 14(2):163–173, 1989.

[7] V. Dahl and M. McCord. Treating coordination in logic grammar. *American Journal of Computational Linguistics*, 9(2):69–91, 1983.

[8] Bonnie J. Dorr and Terry Gaasterland. Selecting tense, aspect, and connecting words: Generating from a temporal database. In *30th Annual Conference of the Association for Computational Linguistics*, 1992, submitted.

[9] Gazdar et al. *Linguistic Inquiry*, 13(4), 1982.

[10] T. Gaasterland. *Cooperative Answers for Database Queries.* PhD thesis, University of Maryland, Department of Computer Science, College Park, 1992. Dissertation work in progress.

[11] T. Gaasterland, P. Godfrey, and J. Minker. An Overview of Cooperative Answering. *Journal of Intelligent Information Systems*, 1992. To appear.

[12] T. Gaasterland, P. Godfrey, and J. Minker. Relaxation as a Platform for Cooperative Answering. *Journal of Intelligent Information Systems*, 1992. To appear.

[13] T. Gaasterland, P. Godfrey, J. Minker, and L. Novik. CARMIN: A Cooperative AnsweRing MetaINterpreter. Technical Report to appear, University of Maryland, Department of Computer Science, College Park, Maryland 20742.

[14] T. Gaasterland and J. Minker. User needs and language generation issues in a cooperative answering system. In *ICLP Workshop on Advanced Logic Programming Tools and Formalisms for Language Processing*, Paris, France, June, 1991.

[15] A. Gal. *Cooperative Responses in Deductive Databases.* PhD thesis, University of Maryland, 1988.

[16] H. Gallaire and J. Minker, editors. *Logic and Databases.* Plenum Press, New York, April 1978.

[17] Norbert Hornstein. *As Time Goes By.* MIT Press, Cambridge, MA, 1990.

[18] J. M. Janas. On the feasiblity of informative answers. In H. Gallaire, J. Minker, and J.M. Nicolas, editors, *Advances In Database Theory: Volume 1*, pages 397–414. Plenum Press, 1981.

[19] S.J. Kaplan. Cooperative responses from a portable natural language query system. *Artificial Intelligence*, 19:165–187, 1982.

[20] Robert Kowalski. *Logic For Problem Solving.* Elsevier Science Publishing Co, Inc, 1979.

[21] K. McCoy. Reasoning on a highlighted user model to respond to misconceptions. *Computational Linguistics*, 14:52–63, September 1988.

[22] K. McKeown. Generating natural language text in response to questions about database queries, 1982. Ph.D. Dissertation. University of Pennsylvania.

[23] J.R. McSkimin and J. Minker. Predicate Calculus Based Semantic Network for Question-Answering Systems, 1979.

[24] C. Mellish. Generating natural language explanations from plans. In L. Sterling, editor, *The Practice of Prolog*, chapter 6, pages 181–223. MIT Press, 1990.

[25] J. Minker and A. Gal. Producing cooperative answers in deductive databases. In P. Saint-Dizier and S. Szpakowics, editors, *Logic and Logic Grammar for Language Processing.* L.S. Horward, Ltd., to appear, 1990.

[26] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28, 1988.

[27] A. Motro. SEAVE: A Mechanism for Verifying User Presuppositions in Query Systems. *ACM Transactions on Office Information Systems*, 4(4), October 1986.

[28] M. Pollack. Generating expert answers through goal inference. Technical report, SRI International, Stanford,CA, October 1983.

[29] P. Saint-Dizier. Etude et realisation de esope. Technical report, Universite de Rennes, France, 1983.

[30] C. Shum and R. Muntz. Implicit Representation for Extensional Answers. In Larry Kershberg, editor, *Expert Database Systems*, Tysons Corner, VA, 1987.

[31] R. Snodgrass. Research concerning time in databases: Project summaries. *ACM SIGMOD Record*, 15(4):19–39, 1986.

[32] W. Swartout and J. Moore. Explanation in expert systems: A survey. Technical Report ISI/RR-88-228, Information Sciences Institutute, USC, Marina del Ray, California, December 1988.

Session A8: SOFTWARE ENGINEERING

Session Chair: Mike Demasie

SOFTWARE REENGINEERING

Ernest M. Fridge III
Deputy Chief, Software Technology Branch/PT4
NASA/Johnson Space Center
Houston, Texas 77058

Co-Authors:

Jim Hiott
Senior Software Engineer
Paramax Systems Corporation
M/C U08A, 600 Gemini
Houston, Texas 77058

Jim Golej
Group Leader
The Mitre Corporation
1120 NASA Road 1
Houston, Texas, 77058

Allan Plumb
Project Engineer
Barrios Technology, Inc.
1331 Gemini
Houston, Texas 77058

ABSTRACT

Today's software systems generally use obsolete technology, are not integrated properly with other software systems, and are difficult and costly to maintain. The discipline of reverse engineering is becoming prominent as organizations try to move their systems up to more modern and maintainable technology in a cost effective manner. The Johnson Space Center created a significant set of tools to develop and maintain FORTRAN and C code during development of the space shuttle. This tool set forms the basis for an integrated environment to reengineer existing code into modern software engineering structures which are then easier and less costly to maintain and which allow a fairly straightforward translation into other target languages. The environment will support these structures and practices even in areas where the language definition and compilers do not enforce good software engineering. The knowledge and data captured using the reverse engineering tools is passed to standard forward engineering tools to redesign or perform major upgrades to software systems in a much more cost effective manner than using older technologies. The latest release of the environment was in February 1992.

INTRODUCTION

Programs in use today generally have all of the functional and information processing capabilities required to do their specified job. However, older programs usually use obsolete technology, are not integrated properly with other programs, and are difficult to maintain. Reengineering is becoming a prominent discipline as organizations try to move their systems to more modern and maintainable technologies. Johnson Space Center's (JSC) Software Technology Branch (STB) is researching and developing a system to support reengineering older FORTRAN programs into more maintainable forms that can also be more readily translated to a modern language such as FORTRAN 90, Ada, or C. This activity has led to the development of maintenance strategies for design recovery and reengineering. These strategies include a set of standards, methodologies, and the concepts for a

software environment to support design recovery and reengineering.

This document provides a brief description of the problem being addressed and the approach that is being taken by the STB toward providing an economic solution to the problem. A statement of the maintenance problems,the benefits and drawbacks of three alternative solutions, and a brief history of the STB's experience in software reengineering are followed by the STB's new FORTRAN standards, methodology, and the concepts for a software environment.

STATEMENT OF THE PROBLEM

Based on trends in the computer industry over the last few years, it is clear that computer hardware, languages, and procedures are not static. The software industry recognizes that a large existing software base must be dealt with as new software engineering concepts and software technologies emerge. The old systems use outdated technology and are costly to maintain. At JSC, as in industry at large, there is a large investment in existing FORTRAN software. These FORTRAN systems do not consistently use modern software practices that can increase maintainability. Yet these systems must be maintained for perhaps the next 20 years. Management is seeking ways to reduce maintenance costs.

In the 1960s-70s many FORTRAN programs were developed at JSC, each with its own sizeable software development team, and its own input/output format. These programs could not communicate readily and eventually were "wired" together in a very crude semblance of integration. Standards could not be enforced because FORTRAN did not enforce them and some were not visible by just looking at the code. The problem was aggravated by the lack of training of new developers plus a 50 percent turnover in the very large development staff every two years. In addition, the user organizations had more people doing development than the development group, and these other organizations were not always aware of the standards and support tools available. This history has left JSC with the following problems:

- Many programs are large and difficult to understand, resulting in maintenance problems.

- The problems in maintenance led to users keeping their own versions of programs, resulting in tremendous duplication.

Many of the FORTRAN programs have already been converted from their original dialect of FORTRAN to the FORTRAN 77 standard. Additional conversions will periodically be required even if only to new FORTRAN standards. It is necessary to consider the question, where will that code have to be in five or ten years? Three possible answers come to mind:

- FORTRAN 77 is the current standard, but this will be replaced by FORTRAN 90. As vendors stop supporting FORTRAN 77, existing FORTRAN will have to move to the new standard or to another language.

- Much of the code may move to the Ada language. This will be particularly true on Space Station Freedom work.

- With C being the language of choice for Unix and the X Window System, some of the code might move to the C language.

ALTERNATIVE SOLUTIONS

Three alternative solutions to the problems identified above have been identified: complete redevelopment of the program, code translation to a more modern language or version of a language, and reengineering. Each of these is illustrated in figure 1 and discussed briefly in the following paragraphs.

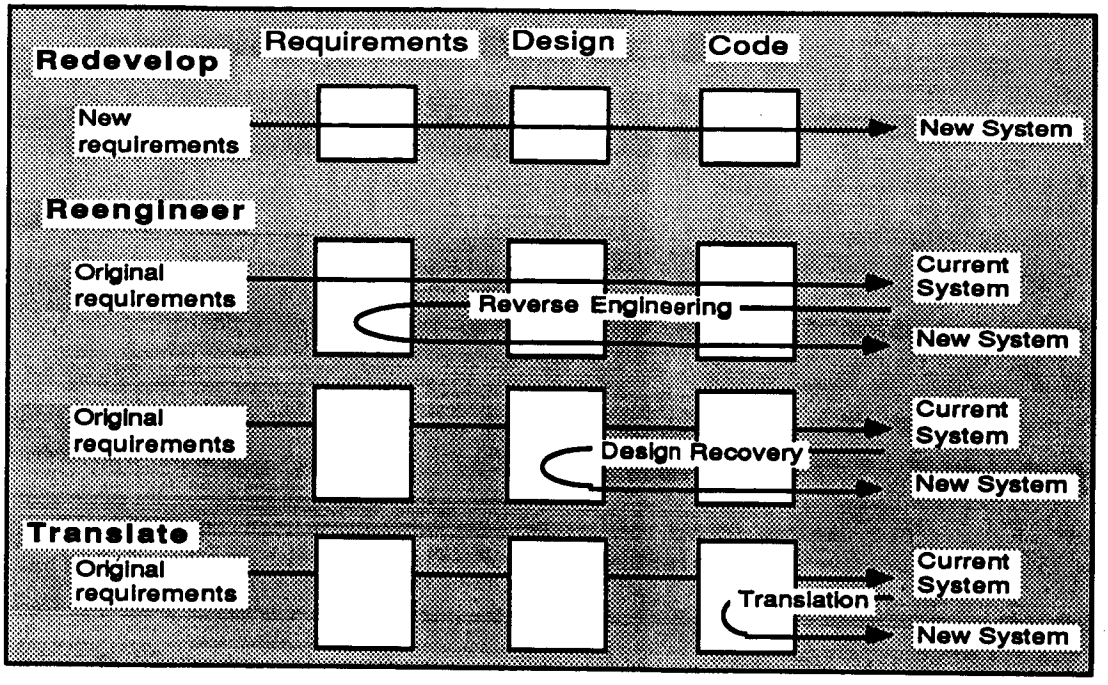


Figure 1. Alternative Solutions

Redevelopment of a system from scratch is very expensive. Redevelopment includes all of the same phases of the life cycle as new development, from requirements through integration and testing. Extensive domain analysis is required, and there is a risk of incomplete requirements. All too often it is claimed that a large program will be redeveloped from scratch to a more modern style only to find out that the new developers did not understand all of the functions and necessary information requirements of the existing system.

Code translation, especially automatic code translation, costs much less. Some might then ask, why worry about all of this now? We can use a translator when the time comes that we are forced to move the code forward. Although this would be a nice solution, the truth is that code translators have proven unsuccessful due to several major reasons:

- Poor existing control flow is translated into poor control flow.
- Poor existing data structures remain poor data structures.
- Input/output translation usually produces hard to read "unnatural"code in the new language.
- Translation does not take advantage of the code and data packaging techniques available in the newer languages. Attempts to automatically translate some FORTRAN programs to Ada have failed.

Reengineering is the combination of "reverse engineering" a working software system and then "forward engineering" a new system based on the results of the reverse engineering. Forward engineering is the standard process of generating software from "scratch." It is composed of the life cycle phases such as requirements, architectural design, detailed design, code development, testing, etc. In each phase, certain products are required and the activities which produce them are defined. Each product is required to be complete and consistent. To progress forward to a new phase normally requires a new representation of the products which involve more detail such as new derived requirements, design decisions, trade off evaluation between alternative approaches, etc. Finally, code is developed which is the most complete, consistent, and detailed representation of the required product.

Reverse engineering is the opposite of forward engineering. It is the process of starting with existing code and going backward through the

software development life cycle. Life cycle products are, therefore, obtained by abstracting from more detailed representations to more abstract ones. This process should proceed much faster than forward engineering since all of the details required are available. Reverse engineering starts with the most detailed representation, which has also proven to be complete and consistent since it can currently do the job required. Developing products in reverse involves abstracting out only the essential information and hiding the non-essential details at each reverse step.

How far to go backward in the reverse engineering process before it is stopped and forward engineering begins is a critical question and involves trade offs. It is important to understand all of what the program does, all of the information it handles, and the control flow since these are probably required to get the job done. This implies taking the reverse process far enough to understand *what* the "as is" program is. This is usually more significant than *how* the program does its job since the *how* is usually the part that will be changed in any following forward engineering process.

What a program does is called its *requirements*. *How* it meets those requirements is its *design*. For a reverse engineered program it is the design that will be updated more often than *what* the program will do. Modern software engineering techniques and technologies such as user interfaces, database management, memory utilization, data structuring, packages, objects, etc. will affect the design, not *what* the program does. Therefore, once it is understood what the program does and what is obsolete, then the forward engineering process can begin with confidence.

Reverse engineering is referred to as "design recovery" when the reverse engineering process stops at the recovery of the design of the implementation, rather than proceeding on to a higher level of abstraction to include the recovery of the requirements. The basic process of this level of design recovery involves recovery of information about the code modules and the data structures in an existing program. This information will support the programmer/analyst who is maintaining an unfamiliar large FORTRAN program, upgrading it for maintainability, or converting it to another target language.

However, a better job of redesigning a program can be accomplished with requirements recovery than with design recovery. To carry the reverse engineering process beyond design recovery to requirements recovery is difficult and requires higher levels of domain knowledge to do the abstractions. The *whys* of the requirements, design, and implementation can only be provided by someone very familiar with the program and the domain. This level of expertise is often very difficult to find and have dedicated to the reengineering process. For this reason, the methods and tools that the STB has developed initially assume reverse engineering only to the design recovery stage. Future development will be based on feedback from the JSC software engineering community. The current standards, methods, tools, and environment are all designed to be sufficiently flexible and extendible to enable the strategies to be extended to cover the full spectrum of reverse engineering.

The overriding philosophy of this planned reverse engineering process is to capture the total software implementation in an electronic form. This includes source code, documentation, databases, etc. Figure 2 illustrates the progression of data structures from COMGEN-compatible code (see section "Software Technology Branch's Reengineering History") to reengineered code. This progression in electronic form ensures that the total consistent and complete requirements representation is available. Software tools are provided to support the generation of the more abstract products required for engineering in reverse as well as capturing rationale and decisions of the engineer. By the continuing process of abstracting the information about the program into the different representations, the engineer can remain more confident that information is not being lost or inadvertently "falling through the cracks."

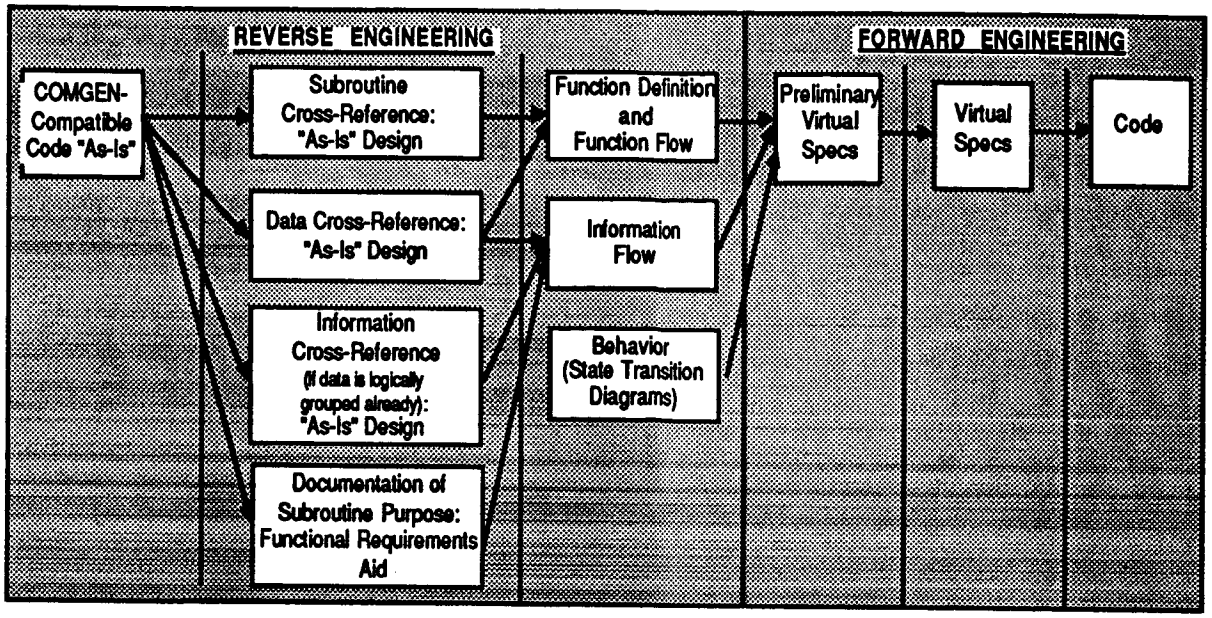


Figure 2. Data Structure Progression

SOFTWARE TECHNOLOGY BRANCH'S REENGINEERING HISTORY

In the early 1970's, the Mission Planning and Analysis Division's (MPAD) Software Development Branch and TRW/Houston developed a tool, called COMGEN, that began as a COMMON block specification statement generator. It grew to include many other functions as new techniques were developed. Later COMGEN was broken up into a continually evolving set of tools with common data interface structures. This tool set supports the maintenance of FORTRAN programs today on Unisys and multiple Unix systems. People still refer to this tool set as COMGEN tools, and a program that complies with the MPAD standard COMMON concept as a COMGEN-compatible program. [1,2,3]

In the 1970's, MPAD performed a lot of software reengineering to meet the goal of combining many of the independently developed engineering programs, each with its own input/output formats. Many of the modern concepts such as separation of input/output processing from the applications, databases, data structures, packages, generics, objects, etc. were recognized and simulated to some degree. They were not called by the modern names, of course, but the design engineers were trying to do good engineering, modularization, and data handling. Even though these techniques were known in the 1970's, they are just now really becoming popular because of newer technologies such as database management systems, user interface tools sets, and modern languages that actually embed and enforce good software engineering practices.

In the late 1980's, some of the personnel and the functions of the Software Development Branch were reorganized into the newly created Software Technology Branch (STB). The STB's reengineering history has put JSC in a better position with respect to the maintainability of its older software than many other organizations. The positive results of this experience include the following:

- Most of the software is reasonably modular.
- The data has some structure.
- Most of the software at JSC is reasonably compatible with the STB's tools, including the in-line documentation.
- The large complex programs that support many simulations have considerable software reuse and information sharing.

MAINTENANCE STRATEGIES

The strategies presented in this document are intended to help with design recovery in support of programmer/analysts who are required to maintain large FORTRAN programs that they did not develop. In addition, these strategies are intended to support reengineering of existing FORTRAN code into modern software engineering structures, which are then easier to maintain and which allow a fairly straight forward translation into other target languages. The STB is proposing standards, methods, and an integrated software environment based upon the significant set of tools built to develop and maintain FORTRAN code for the Space Shuttle. [4,5,6,7,8] The environment will support these structures and practices even in areas where the language definition and compilers do not enforce good software engineering practices.

New FORTRAN Standards

New standards, which allow modern software engineering constructs to be used in FORTRAN 77, have been defined by the STB. [5] These standards are added to existing standards defined by the former MPAD and still in use in the mission planning and analysis domain. The goal of the new standards is to improve maintainability and permit relatively automated translations to newer languages. In table 1, the standards and their benefits are summarized. These standards address documentation, longer variable names, modern control flow structures, grouping subprograms together as virtual packages, data structuring, and input/output encapsulation in separate subprograms. Where FORTRAN 77 does not provide the constructs, virtual constructs are provided along with a tool environment to support their development and maintenance. The existing core of FORTRAN programmers should have little problem with the standards and new FORTRAN code should adhere to them from the start.

Table 1. Standards Summary

Standard	Benefit
Documentation	
Header statement before code blocks	Understandability
Requirements in CD1 statements	Understandability and traceability
Rationale in CD7 statements	Design knowledge capture
Virtual package identification	Maintenance
Longer, more meaningful variable names	Understandability
Modern control flow structures	Maintenance and understandability
Block DO	
DO WHILE	
Grouping subprograms into virtual packages	Higher level of abstraction, understandability
Data structuring	
Preferred use of calling parameters	Maintenance
Controlled use of COMMON blocks	Maintenance
	INCLUDE
	COMMON database concept
Preferably encapsulate input/output in separate subprograms	Maintenance and support to future conversions

Design Recovery and Reengineering Methodology

The reengineering methodology defines the steps, the skills required, and guidelines on how far to reverse engineer before deciding to rebuild. The key goal is to update to modern technology and

software engineering concepts without losing required functions and data. Methods are provided that have the flexibility to meet multiple levels of conversion, each of which improves maintainability. Figure 3 illustrates five methods. [6] Method 1 converts an arbitrary FORTRAN program to COMGEN-compatible FORTRAN, which provides in-line documentation, data structure, and unique data names within a COMMON structure. Method 2 converts software already in this format to the new "standard" FORTRAN with a more Ada-like structure that is ready for a mostly automated translation by Method 3 to a target language that embeds software engineering principles. Alternatively, COMGEN-compatible programs can be converted directly to a target language like Ada by Method 4. Although it is easier to convert a FORTRAN program when the code already meets the standard COMMON concept, commonly known as COMGEN-compatible, arbitrary FORTRAN can be directly converted to a target language by Method 5.

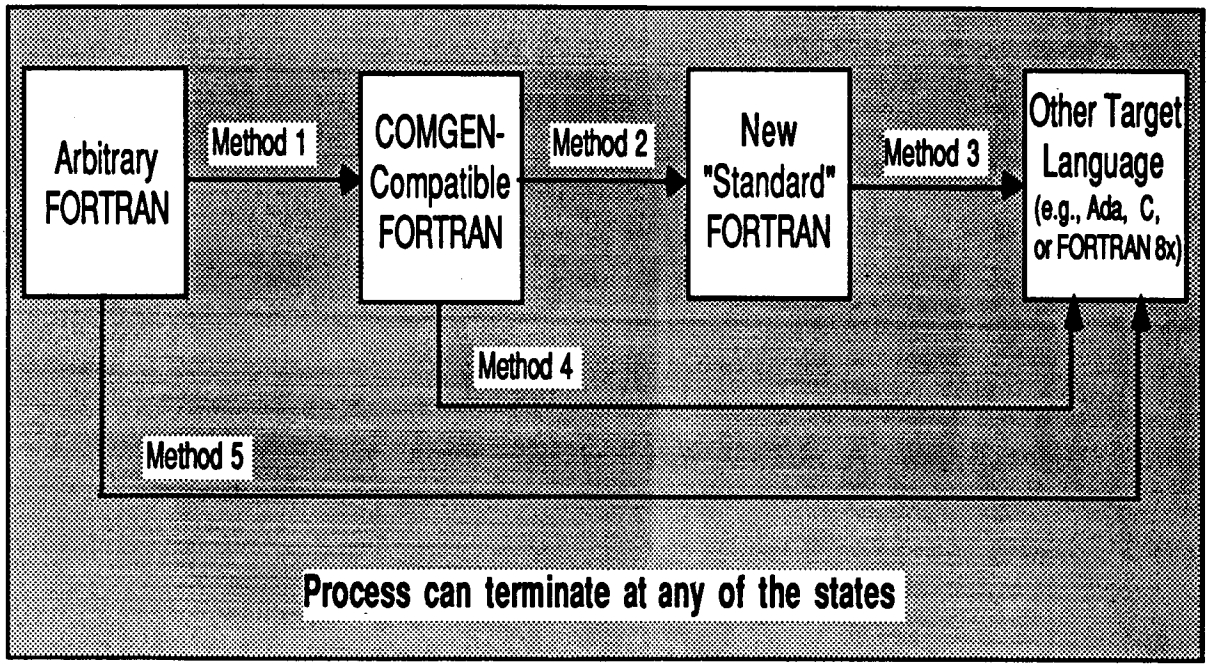


Figure 3. Reengineering Methods

Environment to Support Design Recovery and Reengineering

The STB's reengineering environment [7] is being built around three components: standards, methods, and tools that support the standards and the methods. It contains modified versions of the tools used to support the current JSC FORTRAN programs plus commercial off-the-shelf (COTS) tools and additional custom-built tools. The intent is to get an environment out into use in JSC's maintenance community to provide support for upgrading FORTRAN programs in terms of maintainability in the near-term, then to extend the functionality of the tool set and environment in response to feedback from the programmers/ analysts. Currently several groups at JSC are using the tools. Several tools, both COTS and custom-built, are available for C language support.

The environment has been designed with stable interfaces defined to provide for the maximum feasible degree of seamless integration. It is doubtful that COTS tools can be integrated seamlessly into the environment as no standard interfaces have yet been established for either user interface or data interface (as opposed to data exchange). The tools are integrated at the front end by a user interface and behind the screen by two logical databases, one containing data passed to and from the tools and the other containing the

original and modified source code as shown in figure 4. CASE framework tools are being evaluated as possible integration mechanisms.

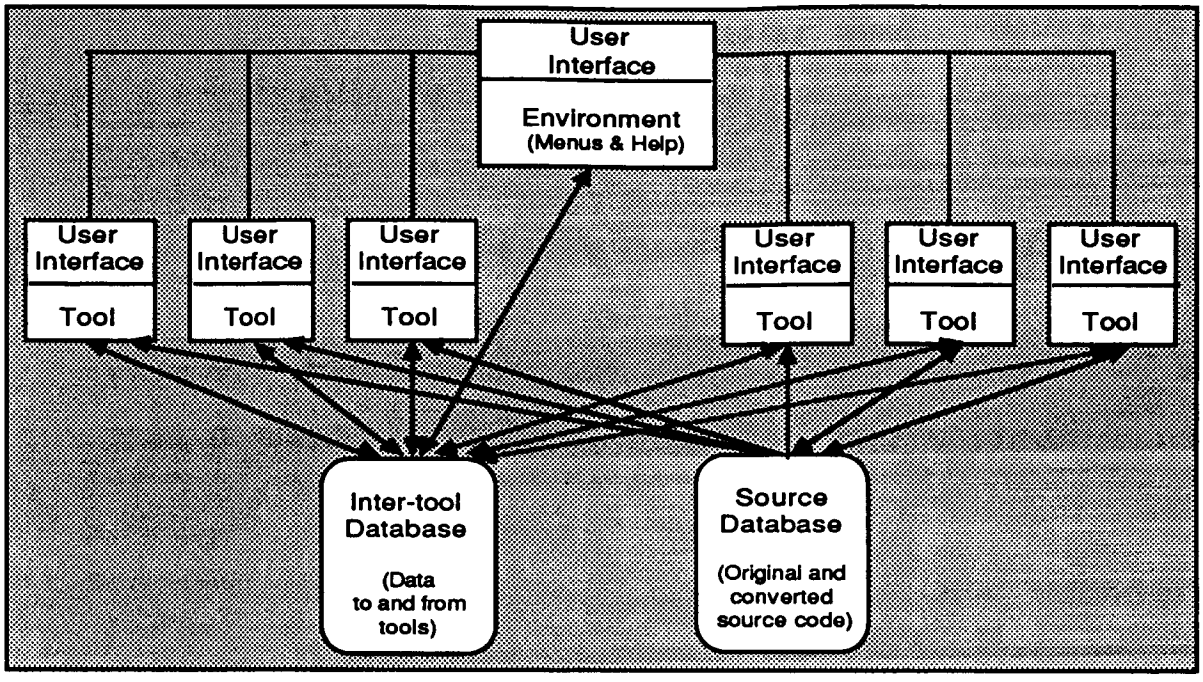


Figure 4. Conceptual Architecture of the Design Recovery and Reengineering Environment

The environment will not be a completely automated environment since much work will still have to be done by a programmer/analyst. A person must be in the loop to provide the required puzzle-solving skills that are beyond the capabilities of state-of-the-practice tools. However, as an experience base is accrued in design recovery and reengineering, knowledge-based capabilities can be added to the environment.

Version 1 of the environment called REengineering APplications (REAP) was delivered in June, 1991. This integrated all existing JSC supported tools discussed above, behind a common user interface built on the MOTIF standard. It contains major elements of all subsystems and encapsulates the capabilities that have been developed and used at JSC during the last fifteen years. A version with improved tool integration, user interface enhancements, and the commercial LOGISCOPE tool was delivered in October, 1991. The FORTRAN design recovery version was delivered in February, 1992. In parallel, the study of using CASE framework standards and tools to better integrate and manage this environment should be completed early in 1992 and the version 2 series will be delivered on one of these platforms. The plans and design of REAP are such, that all deliveries containing COTS products will be tailorable so that users can delete the COTS tools that they do not want to license. This policy even includes the framework integration tools. In most cases, similar functions might still be available but they would have less capability.

CONCLUSIONS

JSC has a large amount of existing code in FORTRAN that embodies domain knowledge and required functionality. This code must be maintained and eventually translated to more modern languages. Three primary alternative solutions have been identified to address the maintenance problems of these old FORTRAN programs: complete redevelopment of the programs, code translation to a more modern language or version of a language, and reengineering. Complete redevelopment is effective but very costly. Simple code translation is cheap, but usually ineffective since seldom do the old systems incorporate modern software engineering concepts such as good data

structuring, good control structuring, packages, objects, etc., that should be present in the new system. Modern languages such as Ada have constructs for representing these features, but translators cannot determine these features in the original code to map them into the new system. Reengineering is being recognized as a viable option because the old systems, in spite of obsolete technology, do contain all of the required functionality and can get the job done. However, at the present time there are only a few expensive Computer Aided Software Engineering (CASE) tools and no total system environment available in the COTS market to support reengineering FORTRAN programs.

The STB maintenance strategies provide standards, methods, and a tool environment for upgrading current FORTRAN systems without losing the embedded engineering knowledge and at a lower cost than for complete redevelopment of the program. A useful environment for reengineering FORTRAN software can be built fairly quickly by building upon the existing FORTRAN development and maintenance tools, COTS products, new software and hardware technologies, plus current research into reuse, design recovery, and reengineering. This environment will support reengineering existing FORTRAN code into more maintainable forms that can also be readily translated into a modern language including newer versions of FORTRAN.

Two versions of the environment were delivered in 1991 which integrate the existing JSC tools plus the commercial LOGISCOPE tool behind a common OSF MOTIF-like user interface. A FORTRAN design recovery capability was delivered in February 1992.

GLOSSARY

arbitrary FORTRAN	FORTRAN program that is not compatible with the COMGEN standards long in place for JSC's mission planning and analysis domain.
COMGEN-compatible	FORTRAN program that is compatible with the COMGEN standards long in place for JSC's mission planning and analysis domain. [1]
design recovery	Reverse engineering, the first step for maintenance or reengineering.
environment	Instantiation of a framework, i.e., an integrated collection of tools. It may support one or more methodologies and may also provide a framework for third party tools.
framework	Software system to integrate both the data and the control of new and existing tools; usual components include a user interface, object management system, and a tool set.
FORTRAN 77	Current ANSI standards for FORTRAN
FORTRAN 90	Future ANSI standards for FORTRAN.
forward engineering	Process of developing software from "scratch," through the phases of requirements, design, and coding.
package	"A collection of logically related entities or computational resources" (Booch[9]).
reengineering	"The examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form" (Chikofsky and Cross

[10]); combination of reverse engineering and forward engineering.

reverse engineering — "The process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction" (Chikofsky and Cross [10]); the first step of maintenance or reengineering; reverse of forward engineering; process of starting with existing code and going backward through the software development life cycle.

software maintenance — Process of modifying existing operational software while leaving its primary functions intact (Boehm [11]).

subject program — Program that is being maintained or reengineered.

virtual package — Package concept as defined by Booch [9], but implemented either in Ada, which enforces the concept, or in a language in which the concept must be supported procedurally.

REFERENCES

[1] Braley, Dennis: *Computer Program Development and Maintenance Techniques.* NASA IN 80-FM-55, NASA Johnson Space Center (Houston, TX), November 1980.

[2] Braley, Dennis: *Automated Software Documentation Techniques.* NASA Johnson Space Center (Houston, TX), April 1986.

[3] Braley, Dennis: *Software Development and Maintenance Aids Catalog.* NASA IN 86-FM-27, NASA Johnson Space Center (Houston, TX), October 1986.

[4] Fridge III, Ernest: *Maintenance Strategies for Design Recovery and Reengineering: Executive Summary and Problem Statement.* Volume 1. NASA Johnson Space Center (Houston, TX), June 1990.

[5] Braley, Dennis: *Maintenance Strategies for Design Recovery and Reengineering: FORTRAN Standards.* Volume 2. NASA Johnson Space Center (Houston, TX), June 1990.

[6] Braley, Dennis; and Plumb, Allan: *Maintenance Strategies for Design Recovery and Reengineering: Methods.* Volume 3. NASA Johnson Space Center (Houston, TX), June 1990.

[7] Braley, Dennis; and Plumb, Allan: *Maintenance Strategies for Design Recovery and Reengineering: Concepts for an Environment.* Volume 4. NASA Johnson Space Center (Houston, TX), June 1990.

[8] George, Vivian; and Plumb, Allan: *A Method for Conversion of FORTRAN Programs.* Barrios Technology, Inc. (Houston, TX), March 1990.

[9] Booch, G.: *Software Engineering with Ada.* Benjamin/Cummings Publishing Co., Inc. (Menlo Park, CA), 1983.

[10] Chikofsky, E. J.; and Cross II, J. H.: "Reverse Engineering and Design Recovery: A Taxonomy." *IEEE Software*, January 1990.

[11] Boehm, B. W.: *Software Engineering Economics.* Prentice-Hall (Englewood Cliffs, NJ), 1981.

N93-32155

AUTOMATION AND HYPERMEDIA TECHNOLOGY APPLICATIONS

Joseph H. Jupin
Edward W. Ng
Mark L. James
Jet Propulsion Laboratory
4800 Oak Grove Drive
Mail Stop 525-3660
Pasadena, CA 91109

Abstract

This paper represents a progress report on HyLite (Hypermedia Library technology): a research and development activity to produce a versatile system as part of NASA's technology thrusts in automation, information sciences and communications. HyLite can be used as a system or tool to facilitate the creation and maintenance of large distributed electronic libraries. The contents of such a library may be software components, hardware parts or designs, scientific datasets or databases, configuration management information, etc. Proliferation of computer use has made the diversity and quantity of information too large for any single user to sort process and utilize effectively. In response to this information deluge, we have created HyLite to enable the user to process relevant information into a more efficient organization for presentation, retrieval and readability. To accomplish this end, we have incorporated various AI techniques into the HyLite hypermedia engine to facilitate parameters and properties of the system. The proposed techniques include intelligent searching tools for the libraries, intelligent retrievals, and navigational assistance based on user histories. HyLite itself is based on an earlier project, the Encyclopedia of Software Components (ESC) which used hypermedia to facilitate and encourage software reuse.

1.0 INTRODUCTION

Space exploration in the 1990's is faced with an information deluge of increasing magnitude and complexities. First, advanced sensor technology has dramatically enhanced our data acquisition capability. As we consider the series of NASA's planetary probes, the Great Observatories, the Earth Observing System, the Space Station Freedom, and the international solar exploration projects, we foresee a rich endowment with large volumes and complexities of scientific and engineering data. Second, the ease of computer publishing, for better or worse, has produced exponentially increasing amounts of documentation, that may include texts, schematics, images, numbers, and even movies for scientific visualization. Third, software continues to grow in amounts and varieties, approaching a phenomenon called "software crisis" by Iwao Toda, Executive Vice President of Nippon Telegraph & Telephone Corp. in Japan. (Ref. IEEE Software, P. 14, May 1992.)

In order to to assist in the management of this information deluge, a system called HyLIte has been proposed as a means of sorting, tracking and managing many of the various pieces of information that when assembled together form the software project. HyLite also expands beyond the typical project management tool by facilitating the incorporation of hypermedia augmented with Artificial Intelligence (AI) techniques to provide even more functionality for the documentation and presentation of information to the user. This paper introduces the various AI technologies that have been selected for implementation into HyLite and the various benefits and justifications for each.

2.0 HYLITE

The Hypermedia Library Technology is a proposed hypermedia system designed for the management, access, and presentation of information in a hypermedia format. Within HyLite will be various tools that will assist the user in the design and implementation of their own hypermedia application. The user will be able to incorporate video, animation, audio and program examples as well the normal hypertext functionality normally associated with hypermedia systems. Though HyLite is still in the conceptual state, a tool which will form the backbone of the system already exists and is in use today. This tool is the Encyclopedia of Software Components (ESC) [1] , which was specifically designed for software re-engineering and reuse. However, as more applications are pursued, variations of ESC will developed that are tailored for different application domains.

2.1 ESC

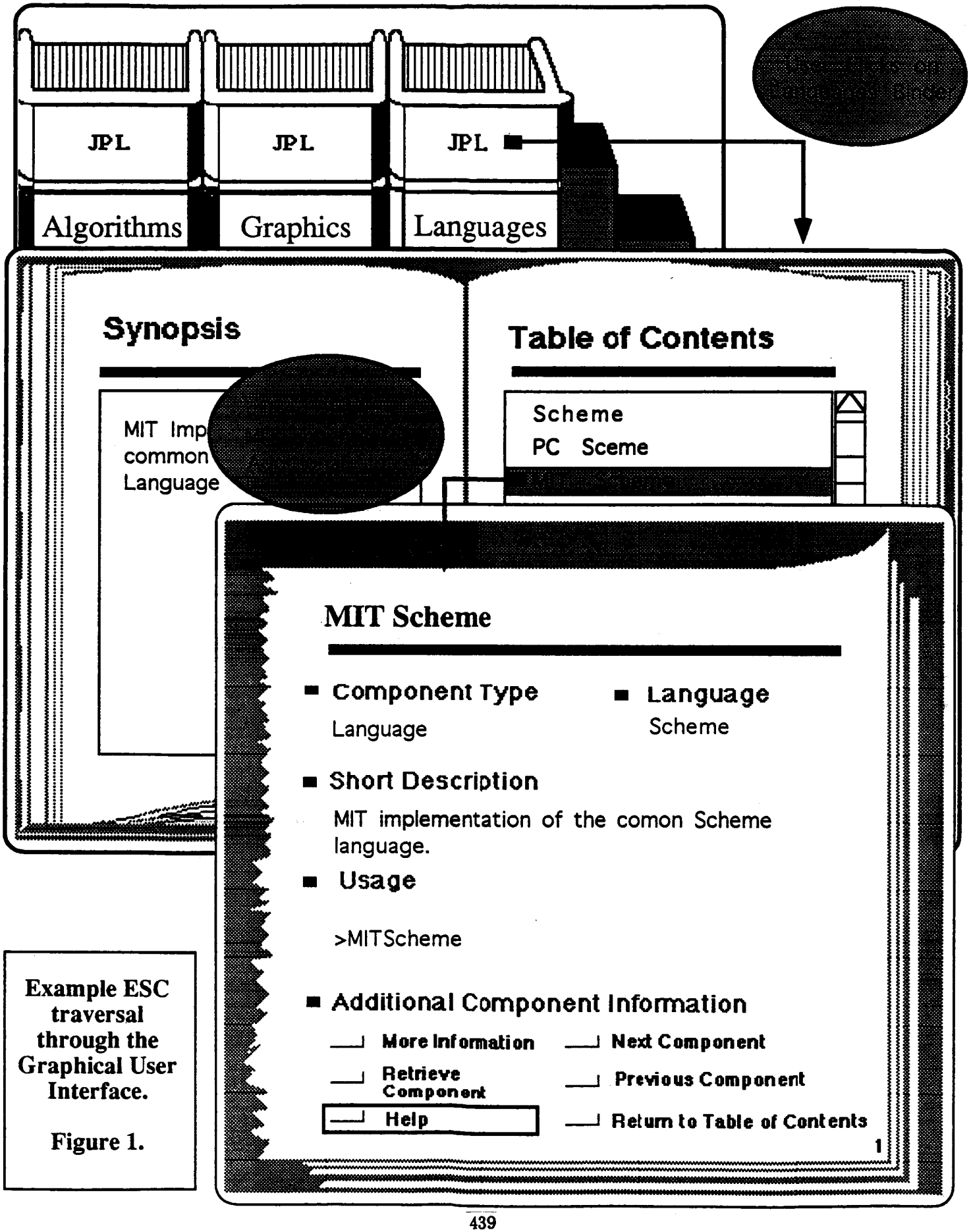
The ESC (Figure #1) is a hypermedia tool designed to encourage software reuse. To this end, many of the features of the tool have been designed to assist the user in locating, retrieving, and browsing for various software artifacts. Much like the encyclopedia metaphor for which the tool is based, a user browses a software encyclopedia looking for various pieces of information (in this case software artifacts) that will assist him in his development efforts. The user is presented with information on each component as he traverses the classification hierarchy. This information includes such items as language type, hardware and software dependencies, animations, graphical representations, and even examples of how the software component is called (calling procedure). All these items allow for the user to arrive at an intelligent decision on whether the software artifact is useful for his project. When arriving at a component that meets the user's requirements, the user merely presses a button and the software is automatically retrieved.

It should be noted that the actual software may or may not reside within a local depository. Therefore, the retrieval mechanism incorporated into the ESC accesses the Internet looking for the requested software artifact. The ESC knows where the software is located based on information within the component description which details not only the software's whereabouts, but also the necessary files needed for the complete package to run. Much of the confusion associated with software retrieval over the Internet is removed from the user's view which adds a level of ease that many users would find appealing.

The previous retrieval was based on the use of the graphical user interface (GUI) representation of an encyclopedia. There also exists within the ESC the ability to execute SQL-like queries on the database to retrieve software. In this case, this feature is designed for advanced users who know the specific item(s) or areas to search for. This functionality is the result of the classification scheme of the ESC (based on semantic networks and frames). A user even has the ability to rearrange the encyclopedia to reflect a specific area of interest (i.e., show volumes based only on language, or graphics, etc).

Another feature of the ESC is the ability to publish new software artifacts as they become available into the ESC so that other users can become aware of their existence. One of the main justifications for the ESC has been the ability to facilitate software reuse by making the search process easier. Therefore, it is imperative that other software artifacts have the capability of becoming a component or volume within the ESC. The publishing engine of the ESC allows a user to describe (classify) the software and submit the information for inclusion into the ESC.

One final feature of the ESC is the ability for the user to develop a user specific version of the ESC that resides on his own hardware. This feature allows the user to create volumes independently on his own hardware and organize and manage the software independently within the ESC. The ESC is a self contained software package that will allow a user to change the encyclopedia's contents to reflect their own interests. Thus, the user can make a personalized version of the ESC.



Example ESC traversal through the Graphical User Interface.

Figure 1.

Given all the functionality of the ESC, it is still important that AI techniques are part of the product so that the software will become even more responsive and flexible for the user. Not only does this benefit the user, but as the ESC is scaled up to become the HyLite system such flexibility and power will be necessary in order for the software to be developed into a fully functional hypermedia authoring/management system.

3.0 INCORPORATION OF AI INTO THE ESC

After the first prototype of the ESC had been completed, research was conducted to determine the feasibility of incorporating AI into the ESC. It was decided that AI could significantly augment the ESC in usability and flexibility. As a result, several areas have been selected for a first pass attempt at inserting different AI tools into the ESC architecture. These areas include navigation assistance for traversing the hypergraph of components, intelligent searchers and intelligent searchers (Figure #2). Each of these have been detailed further below.

- **Intelligent Retrieval**

Currently retrievals must be very specific and based upon predetermined keywords. The keywords reflect the properties that have been used to classify the software and form the connections between the differing components in constructing the classification hierarchy. Thus, the user needs to have a specific idea of the type of artifact needed. In the current system, this is accomplished by the user typing in a keyword (i.e., domain = "Applications") or selecting it from a menu of possibilities. This constraint of keyword driven retrieval needs to be eliminated and a more flexible format adopted.

In response to this problem, it has been suggested that along with the property-component network, there should exist a mechanism for the representation of relationship between properties and components. This would allow a means for intelligent retrievals to occur since the processing engine would be able to infer from the relationships. For instance, a request for Green's Theorem might yield no components from retrieval using the normal keyword search. With the relationship network, the retrieval might suggest other calculus solutions that are approximate to Green's Theorem.

- **Browsing Assistance**

There will be many sessions with the ESC for which the user does not have the context for the desired component. That is, the user is searching blindly for components or even merely casually browsing for anything interesting. To this end, it has been suggested that the creation of a database of user sessions be maintained. From this database, it would be possible for new users to quickly discover relevant components since these paths will have been identified. One possible scenario is one in which a screen pops open displaying a possible candidate at the end of the search path the user is currently navigating. The candidate represents a component that has had multiple visits from other users which translates to a high probability as the object of the user's search.

Another application is specialized sessions based on user login. For instance, if user John Doe consistently browses through the algorithms book of the ESC, then the learning mechanism of the ESC should record this information such that the algorithm book automatically opens for the user as he signs in. Not only does this translate into ease of use for the user, but shows how the system is capable of becoming more responsive to the individual users.

- **Intelligent Spelling Correction**

Spelling errors are a natural phenomena that occur when people interact with a computer. The ESC should be able to correct for common types of spelling mistakes in the context in which they occur. That is if a request is made to retrieve an algorithm for sorting vectors and the word vectors is spelled wrong, then the algorithm should only look for words that are data types that can be sorted. The problem is difficult due to the fact there will be hundreds of thousands of words known by the system. As a result, a spelling correction algorithm cannot iterate over the entire set in a feasible amount of time. In order to delimit the size of the lexicon to be searched, the spelling corrector needs to take into account knowledge about the objects and actions to be initiated. This knowledge if based upon a combination of the classification scheme of the components and an analysis of how actions can be applied to the components (certain items can be sorted, while other items cannot).

- **Overly Specific Requests**

Another problem that will be frequently encountered is an overly specific retrieval request. The user may have a specific idea of what is needed and make an appropriate request. The ESC should be able to process this request not only as a single unit but as sub-sections also. The resulting information will provide the user with more alternatives rather than returning an empty retrieval. That is, if a request is made to retrieve an algorithm that performs a heap sort of two-dimensional arrays and the only algorithm existing within the ESC can sort one-dimensional arrays, then that algorithm should be retrieved with an appropriate message from the ESC. The message should state why the alternate component was retrieved relative to the stated request.

After these tools have been implemented, work is to continue to further research alternate AI technologies for further expansion of the ESC/HyLite's capabilities.

3. CONCLUSION

It is felt that as HyLite and more specifically the ESC gain wider usage, the use of AI tools will significantly facilitate the ease of use of the program. These tools will significantly enhance the retrieval mechanism as well as assist the user in searching for the desired software artifacts. Additionally, the user will gain assistance in the navigation of the large database associated with the ESC. As the ESC transitions over to HyLite, the hypermedia authoring and development system will greatly benefit from AI, also. AI has the potential for incorporation into many differing aspects of the tool ranging from user interface assistance to intelligent file and memory management. Further research is needed to evaluate other potential uses, but initial attempts indicate that the inclusion of AI technology into hypermedia and software engineering tools will significantly enhance the capabilities and ease-of-use of each.

4. ACKNOWLEDGMENTS

We would like to acknowledge Sheldon Shen for his input in this area from a database perspective.

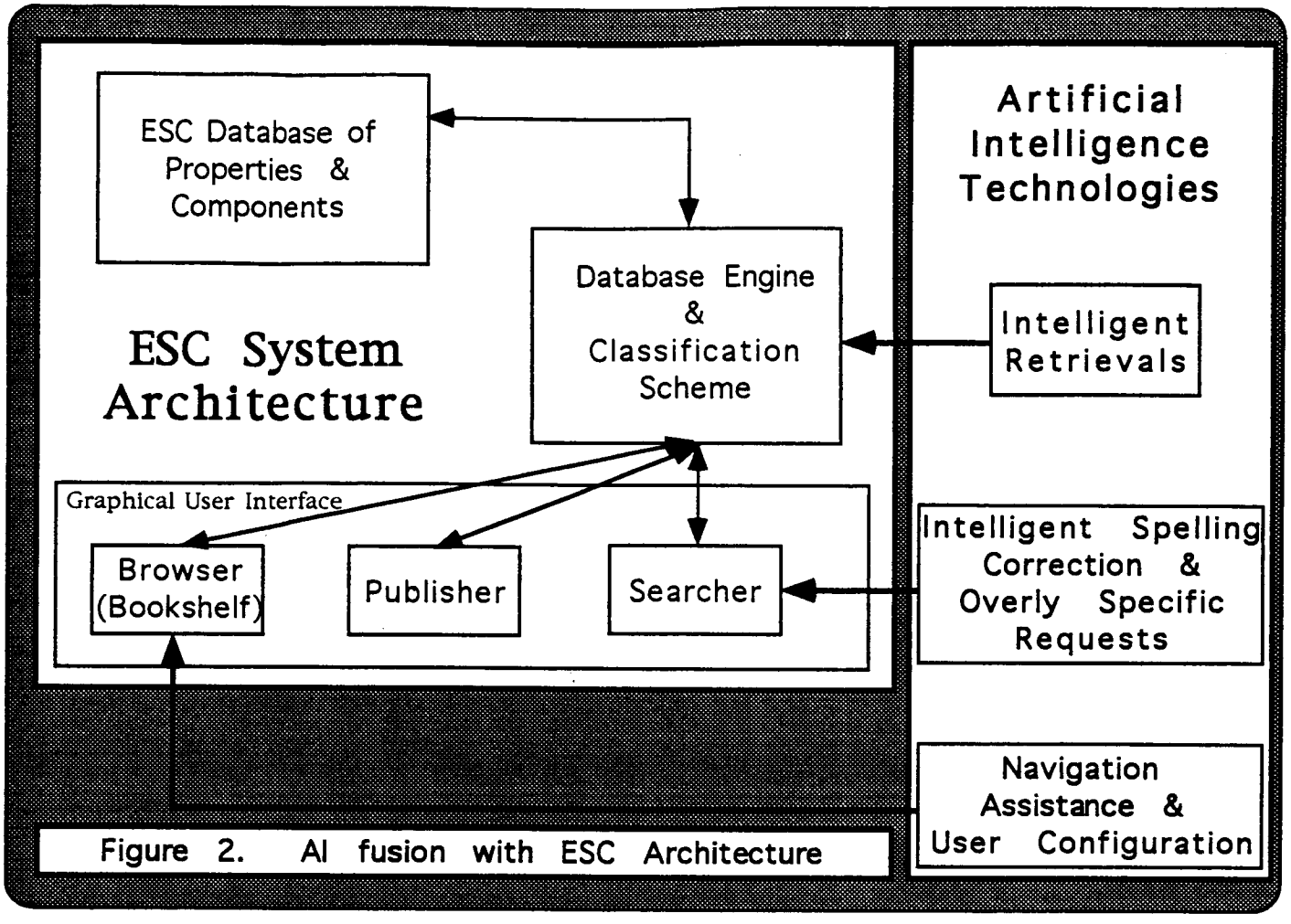


Figure 2. AI fusion with ESC Architecture

5. REFERENCES & BIBLIOGRAPHIES

1. Beckman, B., Jupin, J., Snyder, W. V., Shen, S., Warren, L. V., Boyd, B., Tausworthe, R.C., "The ESC: A Hypermedia Encyclopedia of Reusable Software Components", Information Systems Prototyping and Evaluation Quarterly Report, July 1991, JPL Publication D-8770.

2. Bertsche, S.V. Hypermedia/time-based document (HyTime) and standard music description language (SMDL) user needs and functional specification, X3V1.8M/SD-6.

3. Bullard, L., Price, H., Schoolfield, B., Harlow, R., Dominque, D., Know, E. and Laffin, M. Beyond the Book Metaphor: Concepts for Designing and Implementing Integrated Product Development Environments and Digital Technical Information Services. Obtain from Bruce Schoolfield, GE Aircraft Engines, One Neumann Way MD F126, Cincinnati, OH 45215-6301.

4. Coombs, J.H., Rinear, A.H. and DeRose, S.J. Markup systems and the future of scholarly text processing. Commun, ACM 30, 11 (Nov., 1987).

5. Dykiel, R. Technical requirements for ODA/Hypermedia. ESPRIT project. PODA2, Bull S. A. X3V1.8M/91-2.

6. Englebart, D.C. Knowledge-domain interoperability and an open hyperdocument system. In CSCW 90 Proceedings of the ACM. Available from Bootstrap Project, Stanford University. Doc. (AUGMENT, 132082).

7. Goldfarb, C.F., and Newcomb, S.R. ANSI Project X3.749-D, Hypermedia/Time-based Document Structuring Language (HyTime). X3V1.8M/SD-7 (now ISO/IEC DIS 10744).

8. Markey, B.D. Multimedia and hypermedia standardization, a report from the ad hoc Study Group on Multimedia Standardization to the ISO/IEC's JTC1 TAG. X3V1.8M/90-73.

9. Oak Ridge National Laboratory. Hypertext bibliograph, Apr. 1990. X3V1.8M/90-57.

10. Samuel, R.L., III, Ed. User requirements for hypermedia. X3V1.8M/90-44.

AI FOR SOFTWARE PERFORMANCE TESTING USE OF AI METHODS TO GENERATE TEST CASES FOR SHUTTLE ASCENT SOFTWARE TESTING

Mike Demasie
NASA/Johnson Space Center
2101 Nasa Road 1
Houston, TX 77058

Abstract unavailable at time of publication.

The Knowledge-Based Software Assistant: Beyond CASE

Joseph A Carozzoni
Rome Laboratory: Knowledge Engineering Branch
Griffiss AFB, NY 13441-5700
carozzoni@aivax.rl.af.mil

Abstract

This paper will outline the similarities and differences between two paradigms of software development. Both support the whole software life cycle and provide automation for most of the software development process, but have different approaches. The CASE approach is based on a set of tools linked by a central data repository. This tool-based approach is a data driven and views software development as a series of sequential steps, each resulting in a product. The KBSA approach, a radical departure from existing software development practices, is knowledge driven and centers around a formalized software development process. KBSA views software development as an incremental, iterative, and evolutionary process with development occurring at the specification level.

1. Introduction

Attempts to solve the software crisis have varied from philosophies of management and disciplines of programming to new languages and tools. Many of these innovations have found their way into integrated environments and defined as Computer Aided Software Engineering (CASE). The outcome of these approaches were minor gains in productivity, reliability and maintainability. Although additional improvements may be achieved by continuing in this direction, the order of magnitude improvements needed to address the software crisis are not likely to be realized.

A major problem with present design and implementation activities is an informal development paradigm based on a series of sequential phases. The design rationale involved in the creation of software is lost once the initial implementation is completed. Additionally, modifications to completed systems are performed at the source code level where design information has been obscured by implementation and efficiency considerations.

Recognizing these problems, Rome Laboratory has undertook a program using technology from automatic programming and artificial intelligence to develop a knowledge based system that addresses the entire software life cycle. The Knowledge Based Software Assistant (KBSA), a retreat from pure automatic programming, is based on the belief that by retaining the human in the process, many of the unsolved problems encountered in automatic programming may be avoided. It proposes a new software development paradigm in which software activities are machine mediated and supported throughout the life cycle.

The underlying concept of KBSA is that the software development process will be formalized and automated. This will allow a knowledge base to evolve that will capture the history of the software development process and allow automated reasoning about the software under development. The impact on the software development process is that software will be algorithmically derived from requirements and specifications through a series of user guided formal transformations. Maintenance will occur at the requirements and specification level and the implementation process will be "replayed" as needed.

2. CASE: The Tool-Based Approach

Early efforts to automate software development focused on providing an environment of individual tools designed to handle particular activities. This early approach inspired by DoD 2167 requirements was data and product based. It ignored the software development process. A typical model of an early software development environment is the Unix operating system and a number of associated tools such as text editors, language compilers, the make facility, lint syntax checker, debugger, SCCR version control software, etc. Early tool-based approaches produced modest improvements in software productivity, but tended to ignore the growing crisis in software maintenance. The major drawbacks to this kind of environment were the lack of integration between the tools, and the requirement that the tools be used sequentially. Communication between the various tools relied on the host file system.

Efforts to improve the tool-based approach focused on creating a tighter integration of the individual tools. Reliance on the host file system gave way to more sophisticated "common repositories" such as relational databases. Representative of the improved tool-based approach is Integrated Case (ICASE), Portable Common Tool Environment (PCTE), and Software Engineering Environments (SEE). The primary goal of this evolving paradigm is standardizing the data exchange to allow inter-operability between differing tools and environments. Central to the evolution of the current CASE approach is increased sophistication of the central data repository. Still, mainstream CASE is data and product oriented.

3. KBSA: The Knowledge-Based Approach

While other technologies have readily adapted to production engineering techniques, software has resisted because the creative processes are informal and unspecified. Any solution to the software crisis must address the human intensive facets of software development such as conceptualization and reasoning. Traditional software technology has neglected to address issues dealing with the process of software development. Rather, it has been immersed in addressing the products (ie. written documents, program form, programming languages, management structure, metrics, etc.). Without a paradigm shift from product orientation to process orientation, software development and maintenance will never keep pace with increasing demands.

With this in mind, KBSA has broken away from the traditional tool-based approach in favor of a knowledge based approach to software development. Early attempts at pure automatic programming indicated that the knowledge of programming was still too immature to replace all of the human involvement. With pure automatic programming out of reach, KBSA sought near term relief by keeping the human in the decision making process. Figure 1 is the KBSA process model.

Under the KBSA paradigm, software activities are machine mediated and supported throughout the life cycle. By formalizing and automating the software development process, a knowledge base can evolve that will capture the history of the life cycle processes and support automated reasoning about the software under development. The payoff of this process formalization is that software can be derived from requirements and specifications through a series of human assisted formal transformations. In addition, requirements validation is automatic since the formal transformations have the property of being correctness preserving.

With this new paradigm, maintenance becomes analogous to development and will be performed at the requirements and specification level. Validation and verification are supported as it will be possible to "replay" the process of implementation as chronicled in the knowledge base. KBSA will provide a corporate memory of how objects are related, the reasoning that took place during design, the rationale behind decisions, the relationships among requirements, specifications, &

code, and an explanation of the development process. This assistance and design capture will be accomplished through a collection of life cycle activity facets, each tailored to its particular role. These facets will be highly integrated within a common environment.

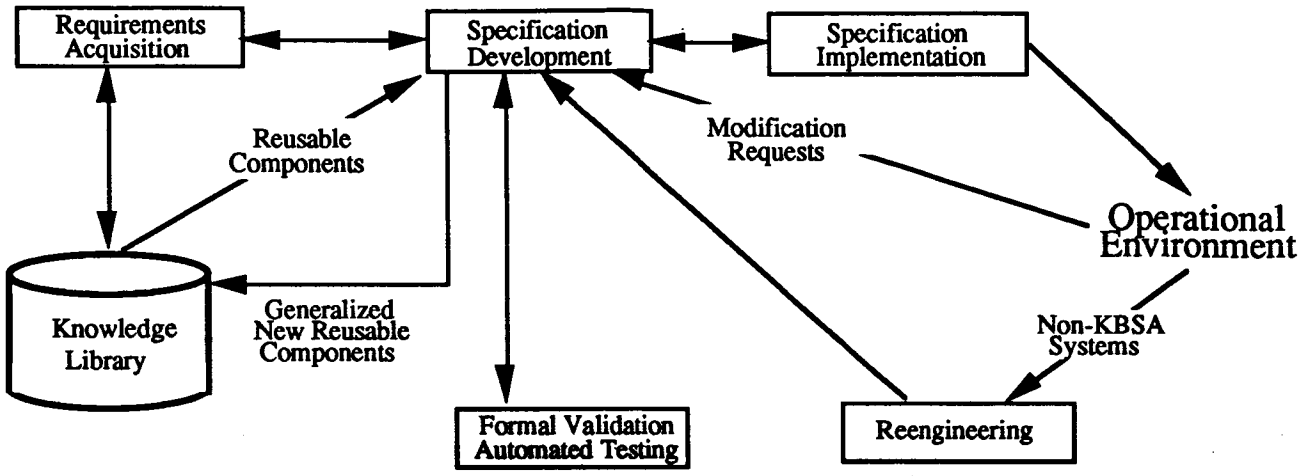


Figure 1: The KBSA Process Model

As envisioned, the KBSA environment will allow design to take place at a higher level of abstraction than is current practice. Knowledge based assistance mediates all activities and provides process coordination and guidance to users, assisting them in translating informal application domain representations into formal executable specifications. The majority of software development activities are moved to the specification level where early validation is provided through prototyping, symbolic evaluation, and simulation. Implementations are derived from formal specifications through a series of automated, meaning-preserving transformations, verifying that the implementation correctly represents the specification.

Post deployment support of the developed application is also concentrated at the requirements/specification level with subsequent implementations being efficiently generated through a largely automated "replay" process. This capability provides the additional benefit of design reuse as families of systems are spawned from the original application. Management policies are also formally stated, enabling machine assisted enforcement and structuring of the software life cycle processes.

The techniques for achieving these goals are:

- Formal representation and automatic recording of all the processes and objects associated with the software life cycle.
- Extensible knowledge based representation and inferencing to represent and use knowledge in the software development and application domains.
- A wide spectrum specification language in which high level constructs are freely mixed with implementation level constructs.
- Correctness preserving transformations that enable iterative refinement of high level constructs (specifications) into implementation level constructs (HOL or machine code) as KBSA carries out the design decisions of the developer.

4. Similarities between KBSA and CASE

Currently, software development is a human-intensive task. KBSA and CASE share the high level goal of facilitating software development by automating much of the work that is presently performed by humans. Both provide automated support for software development by encompassing the entire software development life cycle.
A key area in software development enjoying an increase in appreciation is reuse. If unaltered reuse is not possible, then the next step is assistance in taking something you already have in-hand and modifying it to fit your needs. Reuse has the advantage of being fast and the products mostly debugged. Both KBSA and CASE place significant emphasis on reuse.

Both address the life cycle from various view points which are connected by some form of centralized repository of information. Other areas of commonality of KBSA with some of the more robust and comprehensive CASE environments are automated assistance for:

- Documentation generation (e.g. DoD 2176A).
- Sharing and locking mechanisms for concurrent development.
- Expanded view beyond just software to address the development of systems.
- Graphical interfaces to ease development.
- Consistency checking (pre-defined and user-definable).
- Code and data generation.
- Configuration management.
- Prototyping tools.
- Project management facilities (planning, monitoring, and resources)
- Traceability.
- Testing.

KBSA and CASE have similar goals and attempt to enhance software development. Where they diverge is in their respective orientations (product versus process) and the level to which support is provided.

5. Differences between KBSA and CASE

The major differences between KBSA and CASE originate in the choice between product and process oriented development. Some CASE vendors mistakenly regard their approach as being process oriented, when in fact it is product oriented. Most CASE tools implement some derivative of the Waterfall model for software development This model was developed in the late 1960's in response to the growing software needs of DoD, and can be described by three basic characteristics:

- Software development is a sequence of several discrete stages.
- The product of each stage is documentation which in turn feeds the following stage.
- After each stage is completed it must be validated and verified against the prior stage.

There are many inherent problems in this model which impede software development. These problems occur because the Waterfall model is product based and fails to address the process of software development itself. The basic assumption that software development is a relatively uniform and orderly sequence of development steps is flawed. The extreme sensitivity to task sequence in this model does not provide direct support for modern methods of software development such as rapid prototyping and evolutionary development. Incremental, iterative, and evolutionary design is central to successful software development.

KBSA has put aside the product orientation and has placed focused instead on automating the software development process. A major feature of the KBSA approach is its formalized software development process. With the formalized life cycle, the machine captures all software decisions and provides knowledge-based reasoning assistance. This provides a "corporate memory" of the development history and knowledgeable assistance to humans throughout the life cycle. With this change in focus, KBSA has introduced a paradigm shift from addressing low-level programming problems to addressing high-level knowledge acquisition problems.

CASE has evolved from individual tools targeted for individual activities to an integrated set of tools targeted at those same activities. The accomplishment of ICASE has been to smooth out inter-tool communication. KBSA is a much deeper integration of the software development activities. While integration of CASE has occurred with data, KBSA has also integrated knowledge (about the software development process). The key to the flexibility and robustness offered by KBSA is this highly integrated environment.

KBSA offers a form of prototyping which is more robust than that offered by CASE. Present CASE prototyping is user-interface oriented opposed to the incremental, iterative, and executable design prototyping provided by KBSA. CASE typically has little semantic information which limits the sophistication of the analytic tools and has little automatic support for code generation and reuse. KBSA has rich semantic knowledge about the programming domain and the decisions made during system development which permits sophisticated analysis and has automated support for implementation and reuse.

The diagrammatic tools common in CASE environments use informal specifications and only represent the syntactics of a problem. In addition, the informal specifications used by the individual tools do not allow various views to be related and managed in a conceptually clean and consistent manner. The KBSA knowledge-base integrations varying views (different methodologies) into a single, unified formal representation. This single, unified formal representation captures both the syntactics and semantics of a specification. While CASE developers are typically restricted to a single methodology, KBSA developers can work in a mixed methodology environment. This also provides excellent support for concurrent developers.

Another difference is the manner in which reuse is addressed. CASE addresses reuse at the code level. During the coding phase, a programmer attempts to find a code module which most closely matches the design at hand. Finding a suitable code module for reuse can be time consuming. Even if one is found, modifications may have to be performed to fit the intended design. KBSA addresses at a higher level of abstraction and focuses on design reuse. By concentrating on design, reuse occurs during the requirements and specification phases. Design reuse is more direct and efficient than code module reuse. Code itself is not even part of the KBSA paradigm.

Post deployment maintenance now accounts for 80% of all software costs. The Waterfall model used by most CASE environments performs maintenance at the code level and as a separate process. In KBSA, maintenance is part of the evolutionary development process. Thus maintenance is performed at the requirements and specification level. In addition, this approach lends itself well to support reverse engineering. Legacy software systems can be analyzed and "feed" back into the KBSA process model as specification development. The specifications of legacy software can be recovered integrated with new software developments.

6. What Limits the Capability of a S/W Development Environment?

The KBSA approach to software development and maintenance greatly surpasses present state-of-the-art CASE tools. Some of the technological areas where KBSA excels are:

- The level of integration of the environment.
- The handling of informal requirements.
- The transformation of requirements to specifications.
- The types of specifications which may be used.
- If specifications are automatically maintained consistent with each other.
- The level to which specification consistency is addressed (syntactic versus semantic).
- The manner in which specifications are validated.
- The potential for optimization of the final system.
- Project management functions provided.
- The level to which project management functions are integrated and enforced.
- If the paradigm is product versus process driven.

Level of Integration: The major limiting factor in a software development environment is the level to which the system is integrated. The level of integration can be broken down into three primary levels. Early CASE tools represent the lowest level of integration where only data is shared via the host file system. The tools may or may not be stand alone products. The Unix operating system and its standard set of "bin tools" is a good example. The next level of integration encompasses a set of tools (i.e. ICASE) which communicate through a "common repository", normally a standard relational database. Common to both levels of integration is the reliance on "data" as the inter-tool communication medium. KBSA represents the highest level of integration where not only data, but knowledge (of the software process) is dealt with. Integrating both knowledge and data at the repository level requires a much more sophisticated knowledge representation schema than CASE repositories currently allow. KBSA provides robust inferencing and knowledge representation services.

Requirements Acquisition and Implementation: Eliciting requirements from users is difficult and imprecise. Discrepancies easily arise because the user and the developer communicate from different perspectives. Informal requirements are normally formed out of randomly ordered English text. This informal description of system behavior must be coordinated into more organized and structured requirements. The requirements must then be transformed into more precise and less ambiguous specifications. KBSA provides knowledge-based assistance for requirements acquisition (reference the following paper on ARIES in the proceedings).

Transformation of Requirements to Specifications: CASE informally transforms requirements to specifications using a human mediated conversion. This conversion is imprecise and looses requirements traceability links. KBSA uses formal transformations which have a sound mathematical basis. The transformation is machine mediated, correctness preserving, and does not loose requirements traceability links.

Specifications Types: Specifications may be informal, semi-formal, or formal. Informal and semi-formal specifications do not have a complete mathematical basis and cannot be handled with automated theorem proving techniques. Additionally, informal specifications only allow syntactical analysis and cannot offer semantical analysis. Examples of informal and semi-formal specifications are textual statements of work, pseudo-code, etc. Examples of formal specifications are the formal specification languages Z, VDM, CSP, and temporal logic. KBSA is based on formal specifications, and uses knowledge-based techniques to hide the complexity from the user. The use of formal specifications allow KBSA to perform both syntactical and semantical analysis

Specifications Consistency: The degree to which specifications can be maintained consistent with one another is limited by the underlying representation schema. Informal specifications require human mediated consistency analysis which is time consuming and error prone. The use of formal specifications in KBSA allow machine mediated consistency analysis which is automatic and

precise. Maintaining specification consistency is complicated if the environment allows the use of multiple views (mixing of methodologies). The use of formal specifications allow KBSA to integrate multiple views into a *single formal representation.*

Specification Validation. Specifications that were informally derived must be manually validated against requirements. Because the specifications in KBSA are derived by a formal transformation of the requirements, validation is automatically ensured.

Optimization Potential: Two aspects of optimization are program size and execution speed. The level to which optimization can be achieved is dependent on where and when the optimization effort originated. Normally, there are two opportunities where optimization can be performed. The first chance to optimize is with the specification and/or algorithm itself. The other occurs during coding. While current compiler technology is very good at optimization of code, the optimization of the specification and/or algorithm has not been integrated into CASE environments. KBSA has integrated facilities to address specification and/or algorithm optimization.

Project Management Functions Provided: Project management is often considered a separate activity. The result of using a separate project management system is inaccurate and untimely reporting of data. Additionally, it is easy for programmers to bypass project management efforts by inaccurately reporting data. In KBSA, project management is integrated into the knowledge base. This provides instantaneous and accurate reporting, and eliminates attempts to bypass the system. Also, KBSA is being extended to include knowledge-based estimation and forecasting.

Product versus Process Orientation: Software development has not been adaptable to standard production oriented techniques, thus the actual process of software development itself must be addressed. Product-oriented techniques focus on software development as an assembly line operation consisting of sequential phases. Process oriented techniques address software development as evolutionary and transformational. A common misnomer is to label the "process of generating the individual products" as being process oriented, while the actual process of software development itself remains assembly line oriented. KBSA is truly a process-oriented approach to software development.

7. Conclusions

Both KBSA and CASE are meant to improve software development, yet have significant differences in their approaches. The underpinnings of CASE is a central data repository for inter-tool communication and data sharing. In current tools, this repository is typically a relational database. Most CASE tools, whether following the waterfall, spiral, or comparable methodologies, use a phased approach to software development based on a series of sequential steps. This data driven model views the software development process as the generation of individual products (e.g. requirements document, design document, code, etc.). The central data repository in the present CASE model can store only meta-data (data about data), and facilitates control over access to that data. This product oriented model is insufficient and lacks the expressive power required to successfully attack the software crisis. KBSA has a much deeper and richer representation schema at the repository level. KBSA is a knowledge-based approach which includes not only data, but also contains knowledge of the software development process itself. Table 1 highlights significant differences between KBSA and CASE.

As envisioned, the KBSA environment will allow design to take place at a higher level of abstraction than is current practice. Knowledge-based assistance mediates all activities and provides process coordination and guidance to users, assisting them in translating informal

AREA	CASE	KBSA
Development	Phased	Transformational
Prototyping	User interaction; rapid mock up	Incremental, iterative & evolutionary; executable specifications
Validation	Code against intent	Specification against intent
Specification Implementation	Manual (AD HOC)	Automated (provably correct)
Verification	Structural and functional testing of code modules	Minimal; correctness preserving transformations
Documentation	Only products; design history lost	Automatic and current; history and rationale captured
Development Emphasis	Documentation and coding	Executable specification integrity and system evolution
Maintenance	Code patching	Specification revised, then development replayed

Table 1: CASE versus KBSA

application domain representations into formal executable specifications. The majority of software development activities are moved to the specification level as early validation is provided through prototyping, symbolic evaluation, and simulation. Implementations are derived from formal specifications through a series of automated meaning preserving transformations, insuring that the implementation correctly represents the specification. Post deployment support of the developed application is also concentrated at the requirements/specification level with subsequent implementations being efficiently generated through a largely automated "replay" process. This capability provides the additional benefit of reuse of designs as families of systems that can spawn from the original application. Management policies are also formally stated enabling machine assisted enforcement and structuring of the software life cycle processes.

References

1..Green, C. et al., "Report on a Knowledge-Based Software Assistant," RADC Tech. Report TR-83-195, RADC, Griffiss AFB, NY, Aug, 1983.
2..Jullig, R., et al., "KBSA Project Management Assistant," Final Technical Report, RADC, Griffiss AFB, NY, July 1987, TR-87-78 (two volumes).
3.."Knowledge-Based Specification Assistant," Final Technical Report, RADC Contract F30602-85-C-0221, June, 1988.
4..Larson, A. and Huseth, S., "KBSA Common Framework Implementation," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.
5..Sanders Associates, "Knowledge-Based Requirements Assistant," Final Technical Report, RADC Tech. Report, TR-88-205 (two volumes), Griffiss AFB, NY, Oct., 1988.

Epilogue: For more information, one may attend:

The 7th Knowledge-Based Software Engineering Conference
September 20-23, 1992
Sponsored by Rome Laboratory; In cooperation with ACM, IEEE, and AAAI
McLean Hilton at Tysons Corner; McLean, Virginia
Tel: 315-336-0937 (Barb Radzisz)
EMAIL: kbse7-request@cs.rpi.edu

ARIES - Acquisition of Requirements and Incremental Evolution of Specifications

Nancy A. Roberts
Rome Laboratory /C3CA
525 Brooks Road
Griffiss AFB, NY 13441-4505
EMAIL: nancyr@aivax.rl.af.mil

Abstract

This paper describes a requirements/specification environment specifically designed for large-scale software systems. This environment is called ARIES (Acquisition of Requirements and Incremental Evolution of Specifications).

ARIES provides assistance to requirements analysts for developing operational specifications of systems. This development begins with the acquisition of informal system requirements. The requirements are then formalized and gradually elaborated (transformed) into formal and complete specifications.

ARIES provides guidance to the user in validating formal requirements by translating them into natural language representations and graphical diagrams. ARIES also provides ways of analyzing the specification to ensure that it is correct, e.g., testing the specification against a running simulation of the system to be built.

Another important ARIES feature, especially when developing large systems, is the sharing and reuse of requirements knowledge. This leads to much less duplication of effort. ARIES combines all of its features in a single environment that makes the process of capturing a formal specification quicker and easier.

1.0 Introduction

In 1988, Rome Laboratory funded a joint effort between the University of Southern California Information Sciences Institute and Lockheed Sanders Inc. to develop a project named ARIES (Acquisition of Requirements and Incremental Evolution of Specifications). The goal of the ARIES effort was to build a requirements/specification environment to become part of a larger Rome Laboratory program known as the Knowledge-Based Software Assistant (KBSA). In fact, ARIES was a technical follow-on to two earlier Rome Laboratory efforts, the KBSA Requirements Assistant and the KBSA Specification Assistant.

KBSA is a knowledge-based system that addresses the entire software life cycle. It takes a revolutionary approach to solving the software problem by formalizing and automating the processes of software development as well as the products. The impact of this process formalization is that software will be derived directly from requirements and specifications through a series of meaning-preserving and evolutionary transformations. Transformations ensure that specifications that evolve are correct.

The current requirements-engineering process creates many challenges for the ARIES system to overcome. One problem, especially true in large systems, is managing and coordinating the work of requirements analysts' working alone or in groups on a

project. Once the requirements have been captured, the next question is how to bridge the gap between the initial requirement concepts and the formal and complete specifications. There must also be some validation techniques to ensure the captured specification is correct relative to requirements. Lastly, there is an immense amount of information used in large systems. This poses the question, "Can any part of this work be reused?" All of these problems are addressed in ARIES and are described below.

2.0 The ARIES Environment

ARIES is an environment to assist in requirements engineering. One of the main ideas behind ARIES and KBSA has been to automate some of the more mechanical processes of developing software while leaving the high-level decisions up to the analyst. Thus, the ARIES environment is centered around its knowledge base.

The ARIES knowledge base contains knowledge about domains, reusable requirement/specification components, and descriptions of specific systems being developed. In addition, ARIES provides several tools that interact with the knowledge base and that complete the processes involved with requirements engineering. These processes include acquisition, review, evolution, reasoning, and reuse.

2.1 Requirements Acquisition & Review

The acquisition process begins with an analyst or groups of analysts inputting information about the system to be built. This information includes knowledge about application domains, system components and design processes. The analyst may either type in textual information or input information into graphs ie. data-flow diagrams, information-flow diagrams, hierarchies, taxonomies or spreadsheets. All the information input into ARIES makes up the underlying system representation.

The review process looks at the captured requirements and makes sure that they match up with what the user wants. This process uses many of the same tools used in capturing the requirements, but in reverse. A paraphrasing tool transforms specifications into their respective English descriptions. Diagrams, informal text, and formal specifications are "views" into the underlying system representation. Information in one view may be reviewed by looking at the information in another view. The review process shows the errors made when multiple analysts are working on the system. Inconsistencies are caught early that may have caused larger problems later.

2.2 Evolution

The ARIES environment helps to bridge the gap between requirements and specifications. When building ARIES, the developers took into account the fact that most people don't like to work with formal specifications. ARIES uses unique tools called evolutionary transformations [that help convert requirements into specifications. Transformations are the formal operations of the software engineering process. Evolution transformations, when invoked, modify some aspects of the system while leaving other parts unchanged. The transformations check the specification to ensure that the change is consistent with other decisions and automatically propagates changes throughout the entire specification. ARIES generates specifications in Reusable Gist that can be used for reasoning and execution of simulations.

2.3 Reasoning

Once a specification is captured in ARIES, the next step is to use reasoning techniques to test its validity. ARIES provides for two different types of reasoning: static analysis and dynamic analysis [1].

Static analysis tools look at the current state of the system to find inconsistencies. These tools are provided to analysts in the form of functions that can be invoked on folders or folder components. The most commonly used static analysis tool is called the type checker. It reports errors such as unresolved references, duplicate definitions and type mismatches in expressions.

Dynamic analysis tools help the analyst to get an idea of the system's behavior. The main tool for dynamic analysis is simulation. The simulation facility translates descriptions of the system into an executable simulation. Its purpose is to uncover undesirable behaviors and to ensure the presence of desirable behaviors in the current specification. Simulations are usually run on only a part of the problem at a time to make it easier to see where the inconsistencies come from.

2.4 Structure and Reuse

ARIES structures its knowledge base is into objects called "workspaces" and "folders". A workspace is a separate area where analysts can work on their part of the system. An analyst can have more than one workspace to work on two solutions to a problem or to work on a different domain. Each workspace consists of a set of folders. Folders contain formal and informal information about the system. By organizing the knowledge base into workspaces and folders, information can be shared or kept separate.

ARIES' reuse deals with the reuse of requirements knowledge or domain concepts and is based on reusing information in folders. This means creating specialization hierarchies of certain information. The user has the power to control how much or how little the folders are reused.

ARIES features two different types of reuse: reuse in the same domain and reuse across different domains. A simple example of reuse in the same domain is the inheritance of common terminology. An example of reuse across different domains is the reuse of information between an air-traffic control domain and road traffic control domain. Both types of reuse help to control work duplication and the overall size of the software being developed.

3.0 Conclusion

The ARIES environment makes it easy to enter and modify requirements and specifications through the use of natural language paraphrasing, graphical diagrams and formal specifications. ARIES also provides for the constant checking of the requirements and specifications for consistency. An analyst can review requirements information by switching views to get different perspectives. Reasoning about specifications can be done through the use of tools such as a type checker and executable simulations. The organization of the ARIES knowledge base allows for information and work to be reused, and all the tools combined will aid an analyst in the complicated process of requirements engineering. Thus, the ARIES environment integrated in the Knowledge-Based Software

Assistant promises to make great improvements in solving the software problems of today and the future.

References

1. Harris, David R., W. Lewis Johnson, Kevin M. Benner, Martin S. Feather, "ARIES: The Requirements/Specification Facet for KBSA" Final Technical Report

2. Johnson, W. Lewis, Martin S. Feather and David R. Harris, "The KBSA Requirements/Specification Facet: ARIES", Proceedings of the 1991 Knowledge-Based Software Engineering Conference, pp 53-64.

THE ENHANCED SOFTWARE LIFE CYCLE SUPPORT ENVIRONMENT (ProSLCSE): AUTOMATION FOR ENTERPRISE AND PROCESS MODELING

James R. Milligan
Rome Laboratory
ProSLCSE Program Monitor
RL/C3CB
Griffiss AFB, NY 13441-5700
315-330-2054

James E. Dutton
ISSI
ProSLCSE Program Manager
9430 Research Blvd.
Austin, TX 78729-6543
512-338-5729

1.0 INTRODUCTION

Rome Laboratory, Griffiss AFB, New York, and Electronic Systems Center (ESC), Hanscom AFB, Massachusetts, are joint sponsors of a five year program entitled the *Software Life Cycle Support Environment (SLCSE) Enhancements and Demonstration Program* to develop and support a state-of-the-art Software Engineering Environment (SEE) product for software development and post-deployment support.

SLCSE, pronounced "slice", exists today as an advanced development prototype (completed in late 1989) for the product currently being developed, called *ProSLCSE*, by International Software Systems, Inc. (ISSI), Austin, Texas.

During the year following the completion of SLCSE, the prototype environment was beta tested at three USAF AFLC Air Logistics Centers. In addition, SLCSE underwent several independent evaluations. As a result of the beta tests and evaluations, there was overwhelming agreement that the SLCSE concept was a sound one that could significantly impact the Air Force software process in a very positive way. The beta tests and evaluations, however, also revealed several usability and performance issues that need resolution before SLCSE can be fielded for widespread operational use. ProSLCSE is being developed with these very issues in mind.

To increase performance, usability and portability, the ProSLCSE architecture will migrate from the proprietary VAX/VMS based SLCSE architecture to one which supports a network of heterogeneous POSIX workstations. The ProSLCSE user interface will migrate from the SLCSE character-oriented display (e.g., for VT-100 type terminals) to the X-Windows/MOTIF presentation style. The ProSLCSE repository will migrate toward an ECMA PCTE compliant, client-server architecture. Tool improvements will include, among others, CALS-compliant DOD-STD-2167A automatic document generation, and enhanced life cycle traceability and impact analysis capabilities. Undoubtedly, the most significant improvement in ProSLCSE over SLCSE will be in its enterprise and process management capabilities.

The first product release of ProSLCSE is expected in December 1993. Additional product releases/upgrades will continue during and subsequent to the five year contract (which began August 1991) that will produce a commercial-quality product for use by Government organizations,

Government contractors, and industry. ISSI will also provide comprehensive customer support services such as product installation, user-customized training, on-site software support, mission-specific demonstrations, application consulting, tool/system integration with ProSLCSE, and/or the development of specifically desired ProSLCSE capabilities.

2.0 ProSLCSE

ProSLCSE provides a computer-based *framework* that is used to instantiate *environments* that are tailored to accommodate the specific needs (in terms of process, users, data, and tools) of any software development/support *project*. A ProSLCSE environment supports a total life cycle concept where an integrated toolset is applied during each software development phase (concept exploration, demonstration and validation, and engineering and manufacturing development) and is later transitioned, along with an associated repository containing all accumulated information, to the Post-Deployment Software Support (PDSS) phase. The intent of this concept (as shown in Figure 1) is not only to increase productivity and product quality during the developmental phases, but also to improve the supportability of the product by making available to Software Development Support Activities (SDSAs) all development data and a highly effective PDSS toolset.

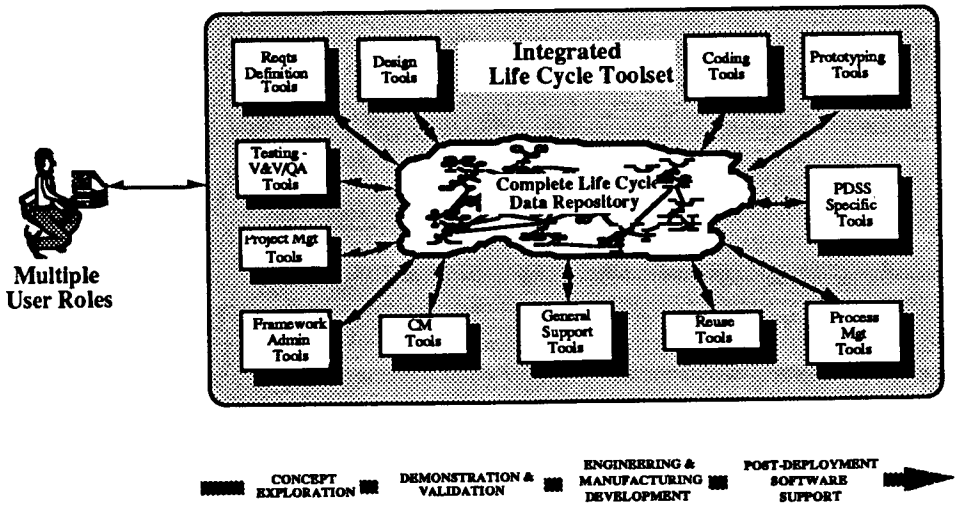


Figure 1 - ProSLCSE: Software System Support from "Cradle To Grave"

2.1 A Process-Oriented Framework

Regardless of the life cycle phase, the operational concept of ProSLCSE always centers around the *process* for a project in which users of the system have jobs to do. The nature of a user's job at any particular time is characterized by the user's *role* which defines the scope of the work that can

be performed by that user (e.g., project management or programming). The roles of users are dependent on the *active* tasks that are currently being performed. User roles, in turn, determine the tools and data that are accessible to users in order to perform those tasks. The user can choose from any one of their *ready* tasks and make it an active task, but cannot perform a *pending* task until all data upon which it is dependent becomes available.

The entire set of tasks, data stores, data flows (dependencies) between tasks/data stores, roles associated with tasks, and personnel-to-task assignments defines the process to be followed for the project in which the user plays one or more roles. ProSLCSE provides the guidance necessary to ensure that users perform task assignments in such a way that does not violate a project's process definition. Figure 2 illustrates a simplified example of this.

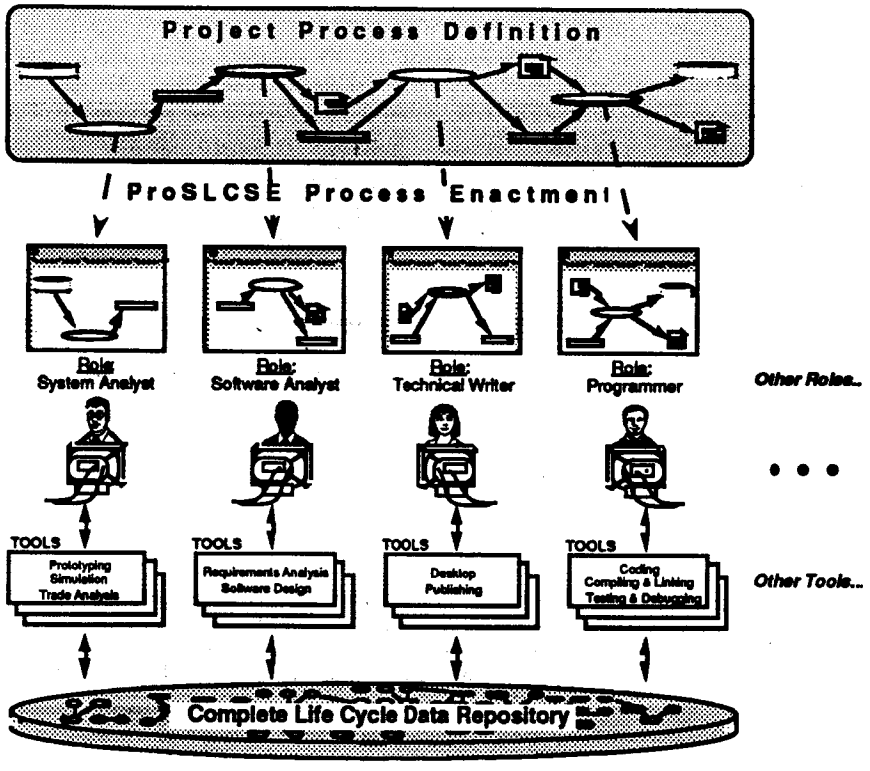


Figure 2 - ProSLCSE: A Process-Oriented Environment Framework

2.1.1 Enterprise Modeling

Enterprise modeling can be defined as the development of the following three submodels:

- Infrastructure Model
- Information Model
- Process Model

While each of these models is conceptually distinct from the other, primarily because it is easier to develop each model independently, the three are far from being physically separate. In fact, ProSLCSE will provide the automated tools necessary to define and tightly integrate all three models within a common life cycle data repository (as discussed in section 2.1.2).

The *E*nterprise and *P*rocess modeling techniques and representations introduced in this paper comprise a methodology known as *FirstEP* (pronounced "first-step"). In FirstEP, each model is defined in terms of a declarative, graphical language [reference to JD's paper] in order to attain the proper mix of expressiveness and ease of use that will enable non-programmers to define complex, real-world situations. The concepts behind each model are now discussed.

2.1.1.1 Infrastructure Model

The infrastructure model describes the people who work in an organization, the tools and other resources they use to perform their jobs, the formal and informal groups to which they belong, their relationships with one another, including reporting relationships, and the communication paths that exist among groups. As shown in Figure 3, infrastructure model concepts include resource, person, tool, machine, location, and group.

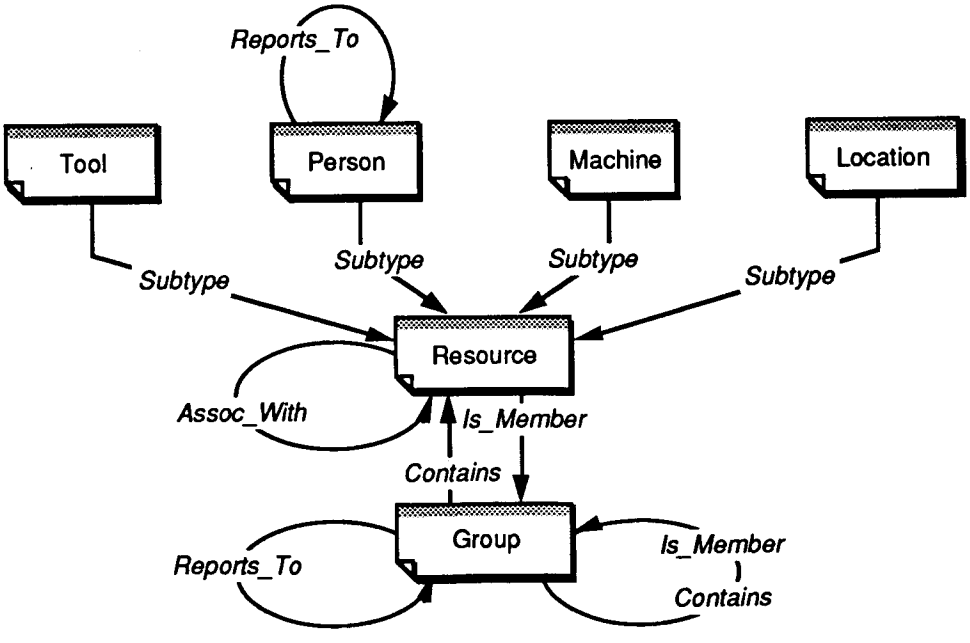


Figure 3 - Infrastructure Model Components

2.1.1.2 *Information Model*

The information model describes the types of objects, such as documents, code, forms, and messages, that are manipulated and exchanged by resources within the organization's infrastructure, as well as the relationships among those object types and the allowed kinds of groupings of objects. As shown in Figure 4, the information model includes the concepts of object, link, object type, link type, attribute, contents, and composite.

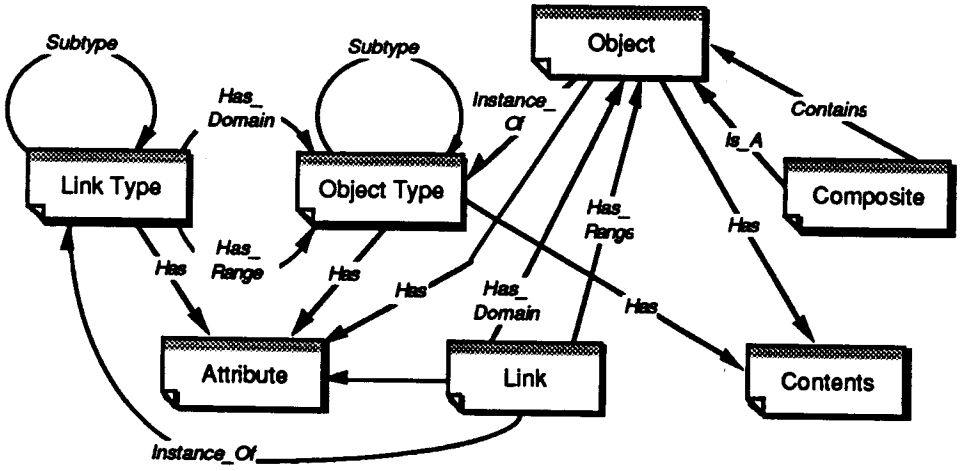


Figure 4 - Information Model Components

2.1.1.3 *Process Model*

The process model describes tasks to be performed, the resources required to perform those tasks, the objects required and produced by each task, and the work breakdown structure in terms of how tasks are refined (decomposed) into lower-level tasks. As shown in Figure 5, the process model includes the concepts of project, process, task, and product.

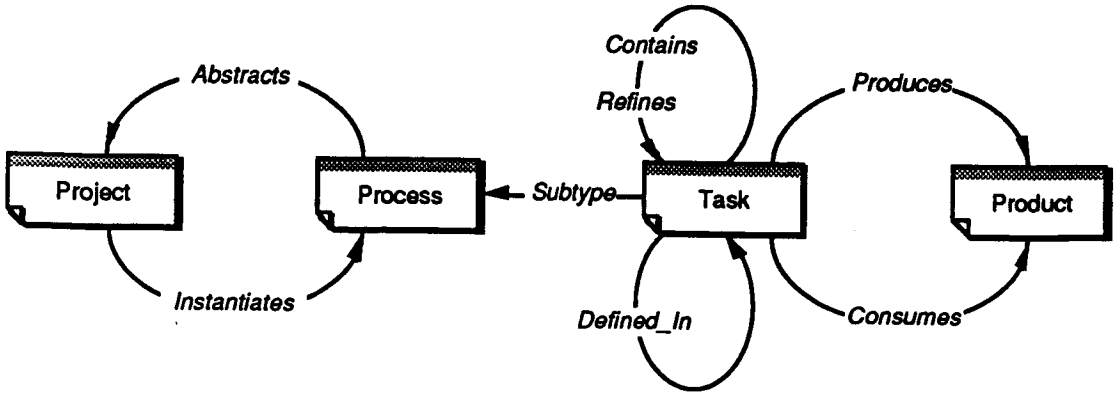


Figure 5 - Process Model Components

2.1.1.4 *Relationships Between Models*

Although each model can at first be defined independently, the establishment of a complete enterprise model comes about only when the relationships between the three models are instantiated. Figure 6 shows some of the relationships between components of each model. The seamless integration of the infrastructure, information, and process models is the key to establishing a controlled environment in which work flow efficiency and quality are optimized.

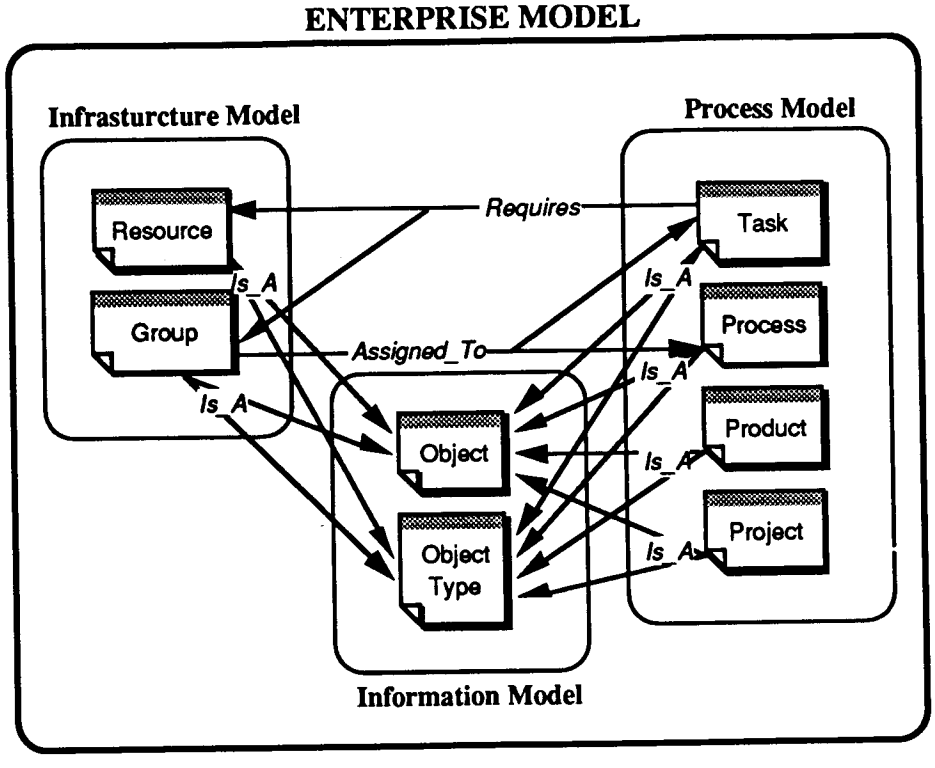


Figure 6 - Inter-Model Component Relationships

2.1.2 *Enterprise Modeling and Process Enactment In ProSLCSE*

A visual process modeling language called *ProVision* will be used in ProSLCSE to facilitate the definition of project processes. Using a ProVision editor, it will be possible to model tasks and communication paths in the form of a directed graph where tasks are represented as nodes, and communication paths (data flow) are represented as edges that connect tasks. Different kinds of communication (e.g., documents, electronic mail, computer input/output, verbal, etc.) are represented by smaller icons along the communication paths.

Similarly, ProSLCSE will provide a visual data definition language editor to define a project's information model, and a visual infrastructure language editor to define an organization's infrastructure model.

Once an environment is instantiated from the enterprise model defined, all model information is stored in a common ECMA PCTE repository. Based on repository information, ProSLCSE will control user work flow, while at the same time, will ease the difficulty of proper task execution by providing a structured work environment to the user.

3.0 CONCLUSION

In this paper, we have introduced a comprehensive method for enterprise modeling that addresses the three important aspects of how an organization goes about its business. FirstEP includes infrastructure modeling, information modeling, and process modeling notations that are intended to be easy to learn and use. The notations stress the use of straightforward visual languages that are intuitive, syntactically simple, and semantically rich.

ProSLCSE will be developed with automated tools and services to facilitate enterprise modeling and process enactment. In the spirit of FirstEP, ProSLCSE tools will also be seductively easy to use.

Achieving fully managed, optimized software development and support processes will be long and arduous for most software organizations, and many serious problems will have to be solved along the way. ProSLCSE will provide the ability to document, communicate, and modify existing processes, which is the necessary first step.

4.0 REFERENCES

1. Dutton, James, "FirstEP: A Common Sense Approach to Enterprise and Process Modeling", ISSI-A92A02001, International Software Systems, Inc., Austin, Texas, June, 1992.

2. Milligan, James, "The Enhanced Software Life Cycle Support Environment: SEE It Today (Tomorrow May Be Too Late)", 3-46, Fourth International Conference on Strategic Software Systems, Huntsville, Alabama, March, 1992.

3. "Proposal for the Software Life Cycle Support Environment (SLCSE) Enhancements and Demonstration Program", ISSI-R9102003, International Software Systems, Inc., Austin, Texas, May, 1991.

4. "Software Design Document for the Enhanced Software Life Cycle Support Environment CSCI of the E-SLCSE System", ISSI-B91A00008A, International Software Systems, Inc., Austin, Texas, June, 1992.

5. "Software Requirements Specification for the E-SLCSE CSCI of the Software Life Cycle Support Environment (SLCSE) Enhancements and Demonstration

Program", ISSI-B91A00007A, International Software Systems, Inc., Austin, Texas, February, 1992.

6. "Statement of Work for the Software Life Cycle Support Environment (SLCSE) Enhancements and Demonstration Program", Rome Laboratory, Griffiss Air Force Base, New York, January, 1991.