NASA-CR-176094
19850023821

NASA SPACE STATION AUTOMATION: AI-BASED TECHNOLOGY REVIEW

March 1985 (Revised April 1, 1985)

By: Oscar Firschein, (Principal Investigator)
Artificial Intelligence Center

Michael P. Georgeff
Artificial Intelligence Center

William Park, Peter C. Cheeseman
Robotics Laboratory

Jacob Goldberg, Peter Neumann,
William H. Kautz, Karl N. Levitt
Computer Sciences Laboratory

Raphael J. Rom, Andrew A. Poggio
Telecommunications Sciences Center

Prepared for: NASA-Ames Research Center
Mountain View, California

Attention: Dr. Henry Lum

Contract No. NAS2-11864

SRI Project 7268

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
(415)859-6200
Telex: 334486

SRI International

NASA SPACE STATION AUTOMATION: AI-BASED TECHNOLOGY REVIEW

March 1985 (Revised April 1, 1985)

By: Oscar Firschein, (Principal Investigator)
Artificial Intelligence Center

Michael P. Georgeff
Artificial Intelligence Center

William Park, Peter C. Cheeseman
Robotics Laboratory

Jacob Goldberg, Peter Neumann,
William H. Kautz, Karl N. Levitt
Computer Sciences Laboratory

Raphael J. Rom, Andrew A. Poggio
Telecommunications Sciences Center

Prepared for: NASA-Ames Research Center
Mountain View, California

Attention: Dr. Henry Lum

Contract No. NAS2-11864

SRI Project 7268

Approved:

Donald L. Nielson
Director, Computer Science and Technology Division

Stanley J. Rosenschein
Director, Artificial Intelligence Center

SRI International®

333 Ravenswood Ave • Menlo Park, CA 94025
(415) 326-6200 • TWX 910-373-2046 • Telex 334-486

This Page Intentionally Left Blank

Pages III and IV Missing

CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGMENTS

Many individuals and organizations contributed to our study, either in informal discussions or by providing draft versions of studies in progress.

NASA

Dr. Victor Anselmo, NASA Headquarters, manager of the SSAS study, and Dr. Henry Lum, NASA Ames, monitor of the SRI study, provided excellent coordination and direction. Other NASA contributors were

NASA Headquarters

Mr. Daniel Herman, Chief Engineer, Space Station Program

Dr. Ron Larsen, NASA OAST

NASA Johnson Space Flight Center

Mr. Aaron Cohen, ATAC Chairman

Dr. Jon D. Erickson	Dr. Owen Garriott	Mr. Lyle Jenkins

NASA Langley Research Center

Dr. Al Meintel, Jr.	Ms. Nancy Orlando	Mr. Jack Pennington
Mr. Jim Wise	Ms. Kathy Abbot	Mr. Steve Millman

NASA Marshall Space Flight Center

Mr. Jon Haussler	Mr. Walter Frost	Mr. Ken Fernandez
Mr. Audie Anderson	Mr. Kieth Clark	Mr. Tom Brian
Mr. Don Scott	Mr. Jim Graves	Mr. Jim Atherton
Mr. Tom Bryan	Mr. Fred Roe	

NASA Ames Research Center

Dr. Ewald Heer	Mr. Vic Vykukal	

Jet Propulsion Laboratories

Dr. Antal Bejczy	Mr. Rick Killion	Mr. Robert Staehle
Dr. Stephen Vere	Mr. Mike Heissen	

NASA Goddard Research Center

Dr. Henry Plotkin	Mr. Lloyd Purves	Mr. Tim Premack
Mr. Stephen Brodd	Mr. Scott Gordon	Mr. Len Solis

Contractor Participants

TRW Space and Technology Group

Mr. Hans Meissinger	Mr. Jeff Spitzer	Mr. Dick Molgaad
	Mr. Bill Palmer	

General Electric Company

Dr. Donald Kugath	Mr. Roy Biddiscome	Mr. Robert Brotherton, Jr.
Mr. Ray Hallett		

Martin Marietta Aerospace Corp.

Mr. Richard Spencer	Mr. Roger Schappell	Mr. Dennis Haley

Hughes Aircraft Company

Mr. James T. Yonemoto

Boeing Aerospace Company

Ms. Amy Toussaint	Mr. Paul Meyer	Dr. Douglas Donough

Automation and Robotics Panel

Dr. David Criswell	California Space Institute
Dr. Charles Rosen	Machine Intelligence Corporation
Dr. Robert Cannon	Stanford University
Mr. Carl Ruoff	Jet Propulsion Laboratory
Dr. Roger Cliff	DARPA

Universities and Industry

Mr. Richard Wallace	Carnegie-Mellon University
Ms. Ping Lee	MIT Space Systems Laboratory
Mr. Russ Howard	MIT Space Systems Laboratory
Mr. Scott Millard	Western Space and Marine, Inc.

SRI International

Dr. Donald L. Nielson	Director, Computer Science and Technology Division
Dr. Stanley J. Rosenschein	Director, Artificial Intelligence Center
Dr. David Nitzan	Director, Robotics Laboratory
Mr. Bruce Webbon	Mr. Les Montgomery

We would like to thank Dr. Savel Kliachko, specialist technical writer/editor at SRI, for his invaluable suggestions, and Ms. Joan K. Ichiki for incorporating them into this report.

OVERVIEW

A. Introduction

As part of the enabling legislation for the space station, Congress requested that NASA establish an Advanced Technology Advisory Committee (ATAC). ATAC was to identify promising automation and robotic technology for the space station, and make recommendations that would comprise an integral part of its definition and preliminary design contract. These recommendations were to be given to Congress by April 1, 1985. NASA established the Space Station Automation Study (SSAS) as a source of informed technical guidance for ATAC in the use of autonomous systems to implement space station functions. Such systems are expected to provide U.S. industry with vital automation capabilities.

The SSAS was conducted by a concept design team and a technology team. Each member of the concept design team examined particular topics relevant to the space station to determine how the required functions could be automated. The corporate members of that team and the topics they reported on were (1) TRW (satellite servicing), (2) GE (space manufacturing), (3) Hughes (subsystem autonomy), (4) Martin-Marietta (autonomous systems and assembly), and (5) Boeing (man-machine interface). The role of SRI, as the technology team, was to utilize the automation concepts postulated by the first four concept teams to determine what research and development would be required in artificial intelligence (AI) and computers to attain the capabilities implied by these concepts.*.

The goals of the SRI study were (1) to provide guidance with respect to the state of the art in artificial-intelligence (AI)-based technologies; (2) review the results of the concept design contractors to determine the AI capabilities required by the designs; (3) delineate a series of demonstrations that would indicate the existence of these capabilities; and (4) develop a research-and-development plan leading to such demonstrations. As a separate issue, advanced techniques for the space station's information system were also to be investigated.

The methodology used in the SRI study consists of the following steps:

(1) Examine automation concepts prepared by the concept design

---

*Our role was not to determine the optimal mix of man and machine in the space station. This topic is covered in [1]

contractors and determine needed automation capabilities.

(2) Derive sequences of demonstrations leading to the desired automation capabilities.

(3) Derive research and development plans leading to technology for carrying out these demonstrations.

We first reviewed the material provided by the concept design contractors and identified the implied automation capabilities required. After determining the latter, we then postulated a series of demonstrations that would verify the existence of these capabilities. Finally, for each of the AI-based technologies, the relevant research and development to carry out the demonstrations are indicated.

B. Summary

The research and development projects in automation technology described in this report can yield the following essential advantages of crew safety, productivity, increased autonomy, and augmented capability that will ensure successful, maximally efficient operation of the space station. Many of the research projects also have extremely promising potential for innovative results that can be applied directly to terrestrial automation.

- Crew safety. Increased astronaut safety through a reduced need for EVA, and the ability to deal safely with malfunctions that cause hazardous conditions in the vicinity of the failed equipment.

- Productivity. Increased astronaut productivity through greater dexterity (compared with suit gloves), reduced space-suit maintenance and EVA overhead (prebreathing time, need for a backup crew member, etc.); less time spent by crew and mission specialists in performing routine housekeeping and station operation tasks such as monitoring, maintenance, and malfunction handling; and a smaller support team needed to provide services to "paying customers."

- Space station autonomy. Decreased cost of ground mission support and increased mission versatility.

- Augmented Capability. Telepresence systems, robots, and perhaps even robot supply tenders could be left in geosynchronous orbit for extended periods to service satellites. Such servicing could also be carried out by remote control from earth, relieving the space station crew of control tasks. There will also be the

> ability to sense, identify, and correct malfunctions either instantly or very quickly. Even the fastest possible human response to onboard subsystem failure—e.g., requiring crew members going into EVA to prebreathe oxygen, get into a suit, egress, and move to the problem area—may be too time-consuming to cope effectively with serious emergencies.

While the purpose of this study was to propose demonstrations and R&D that would indicate the technology needed, we did not estimate the funding levels necessary. However, in this respect, it should be noted that DARPA's Strategic Computing Program (SCP) commitment totals approximately $300 million over the next three years, and that the needs of space station automation identified by the concept contractors are at least equivalent to the tasks comprising the SCP. Thus, if NASA is to derive maximum benefit from space station automation, an investment of at least $100 million per year in research and development is certainly not unreasonable. In particular, if the space station is to serve as a driving force for industrial automation, it is essential that substantial funding be provided for research in advanced automation, especially robotics and artificial intelligence, rather than concentrating exclusively on more immediate engineering issues. We summarize here what is said regarding the need for NASA support for the various automation technology disciplines, taking into account research now being done under other auspices.

1. *Teleoperation/Robotics*

Although research in automation technology is being carried out by DARPA and other agencies, the special needs of space and the concomitant motivation for focusing on a different set of objectives are compelling NASA to initiate projects that are relevant to its purposes and that utilize the available resources and accumulated experience of the scientific community. Specifically, there are unique environmental problems (zero gravity, vacuum, etc.) not found on earth. In addition, for space telepresence equipment to evolve smoothly towards greater autonomy, it must be built with more quality than an industrial robot, yet also be very dexterous. The combining of these two criteria is something new. No equipment on the market meets both requirements very well, and certainly none has been designed from the standpoint of weight minimization and space qualification.

For the greatest possible effect on the progress of the space station, NASA research in telepresence carried out at the various NASA centers should be expanded and coordinated. In particular, intensive early research and development are needed on *telepresence*—with emphasis on slave equipment hardware, work station design, and related software. A sufficiently vigorous effort would produce space-qualified equipment with useful levels of

dexterity—and do so in time to meet the contractors' schedules for their automation concepts.

2. *Sensing*

NASA-sponsored sensor research should concentrate on research and development not included in the DARPA program, i.e., visual and tactile sensors leading to the transition from teleoperation to more automated operation of robot arms. A basic goal of the research should be to develop algorithms and techniques that will achieve automatic interpretation of complex objects under variable lighting conditions. Such a capability is essential for directing manipulator arms and effectors in the execution of a task. A CAD data base often plays a key role in making this kind of interpretation possible. Mobile robots should be able to determine their location by means of easily read fiducial points distributed throughout the space station. Finally, NASA should encourage the development of tactile sensors and algorithms for interpreting the tactile data, since it is specifically this capability that will be needed for sophisticated object manipulation.

3. *Expert Systems*

In examining space station applications, it is evident that a high return on research investment, in terms of safety and effective utilization of ground and spacecraft crew, is to be found in automation of the operation, maintenance and control of space station subsystems and manufacturing processes. The crucial characteristic of these applications is that the domain is dynamic—i.e., it involves reasoning about the effects of sequences of actions and tests that can change the state of the world over time. Moreover, because various subsystems will be operating simultaneously, it is important that the representation be sufficiently rich to enable reasoning about concurrency and subsystem interaction, and that efficient procedures for automatic scheduling and synchronization be developed.

Very little research is being done in this area. Consequently, without NASA support it is unlikely that the technology necessary for automating space station operations could be developed before the end of this century. Furthermore, most of the research issues that arise in representing and reasoning about these applications are also of critical importance in developing intelligent robots. Thus, through concentration on generic formalisms, schemes for representation and reasoning can be devised that would be *eminently* suitable for both areas of application. In addition, such generic research would produce major benefits for terrestrial applications, both military and civilian. Space qualification of new expert systems and reverification of existing ones when changes in other subsystems have been made are also important.

4. *Planning*

Planning in DARPA's SCP concentrates mainly on navigation issues and will have very little influence on the more general forms of task planning required for the space station. In particular, the navigation of robotic devices is radically different in the space station environment, requiring reasoning about a dynamic world cluttered with moving objects, rather than the planning of routes in a relatively static domain that consists of varied terrain, enemy positions, etc. General robotic tasks involving spatial/geometric reasoning (such as repairing a satellite) are somewhat related to the DARPA/Air Force Intelligent Task Automation projects, and there is some research of that type being done in industry. It may be possible to adapt some of these results to the specialized requirements of NASA. There is no significant multiagent research being done in the DARPA projects, yet this topic is of critical relevance to many space station tasks in which multiple robots or persons are engaged. Finally, there is very little in the DARPA projects that is concerned with planning to realize goals or perform tasks in nonnavigation activities—nothing, in fact, dealing with repair, construction, or material transfer, all of which are essential for such space station operations as satellite servicing, construction of assemblies, orbiting maneuvering vehicle operations, and transfer of fuels.

5. *Computers*

Most automation in the space station will require the existence of a new generation of computers. An important impact will be exerted on computer technology by the SCP support in three broad areas: (1) signal processing, (2) symbolic processing, and (3) multi-function machines. The goal in signal processing is to build a system capable of executing one billion or more operations per second by 1986, and one trillion operations per second by 1990. The symbolic processor research and development is aimed at applications in vision, natural language, and expert systems. New optical recording techniques will provide multigigabyte, erasable storage.

Exploitation of such new technology and the need to meet the multiple requirements of the space station for computer reliability and performance will place considerable stress on current technology. It will require architecture that allows rapid integration of new techniques in a way that preserves system integrity and satisfies ever-increasing requirements for performance. The hostile natural environment necessitates a computer design of extraordinary reliability. An integrated model of system data, as well as new approaches to data management and retrieval, must be provided to deal with masses of data of different types.

The Space Station Information System (SSIS) application and operating-system software can currently be designed so as to evolve into more distributed-

processing configurations when they become feasible. These design techniques should be employed in the initial SSIS system. The greatest challenge will be to integrate these techniques with new ones that are emerging from current research, so as to achieve all of the required goals simultaneously. Thus, the SSIS requires new research approaches, new architectural techniques, and better computer system engineering. In our view, the computer research topics that will yield the most benefit are a unified hierarchical-distributed architecture, software engineering approaches that support higher levels of programming abstraction, an intelligent data system that supports a unified model of multitype data, the application of expert system techniques to system integrity, and hierarchical fault analysis and recovery.

6. *Man-Machine Interface*

The incorporation of techniques for automated, but human-supervised, control of large, complex, high-risk systems such as the space station is based on the rationale that this mode of control will provide greater efficiency and reliability than would be otherwise obtainable. However, research is needed on how to display integrated dynamic-system relationships in a way that is understandable and accessible to the human, and how best to allow the operator to tell the computer, in a flexible and natural manner, what is desired and why.

The operator's cognitive process must be aided by computer-based knowledge structures and planning models. Results of DARPA's natural-language and speech research should be utilized for more effective man-machine communication, particularly in situations such as EVA where voice input/output has very special advantages.

It will also be important to develop techniques for coordinating the efforts of the different people involved in supervising the same system. This research should be coordinated between NASA and the various DoD agencies, since all are faced with a similar problem.

C. Conclusions

The challenge of space station automation will inspire advances in AI-based technology, acting as a spur to integrate and focus the combined efforts of diverse disciplines. These accomplishments will make the space station more effective and provide U.S. industry with vital automation skills for the future. Because the space environment brings with it problems not encountered on earth, and because the very survival of the crew depends upon the reliability of the space station, it is essential that NASA ensure its strong support of purposefully directed AI-based technology research.

REFERENCES

1. "The Human Role in Space (THURIS)," (Final MDC H1295), Huntington Beach, California, McDonnell Douglas Astronautics Company, October 1984. Conducted for MSFC.

I INTRODUCTION

Pursuant to President Reagan's directive in the State of the Union address, NASA has begun a program that will result in a permanently manned, fully operational United States space station by the early 1990s. The space station will support scientific and commercial endeavors in space, stimulate new technologies, enhance space-based operational capabilities, and, in general, maintain America's leadership in space during the last decade of this century and into the next. As a first step in carrying out the President's directive, NASA is in the process of defining a space station program (SSP) to accomplish a range of missions to be conducted during the 1990s and beyond. The SSP will comprise various space station program elements (SSPEs) such as space stations, space platforms, and orbiting maneuvering vehicles (OMV).

The space station is to be built for several decades of service—well into the 21st century. It must therefore be designed for future capability insofar as we can now foresee just what that should comprise. Obviously, it must also be designed and implemented in logical steps that evolve toward that capability. Because the ability to use man in a highly productive way aboard the space station is a key objective, an optimum integration of functions must be developed between man and machine. Another basic goal of the SSP is to be "customer-friendly"—i.e., dedicated to fulfilling customers' needs. Some of the requirements that relate to AI-based technology are as follows:

- The space station, together with its systems and subsystems, will have the capability to be progressively modified or upgraded in-orbit to accommodate evolving technologies.

- The SSP shall provide for the servicing and maintenance required by the orbital maneuvering vehicles (OMV) based at the station. The OMV will be utilized for the orbital maneuvers necessary to service co-orbiting satellites or platforms, or to bring those without their own propulsion capability into proximity with the station for servicing.

- As one step in the SSP, an orbital transfer vehicle (OTV) will be utilized for the transfer of payloads to and from higher-energy orbits.

- The initial configuration and later station operations will include

shuttle-tended modes for transfer of equipment and consumables to and from the space station.

- An integral subsystem of the space station will be one or more remotely operated manipulators for a variety of operations. Some of their more critical functions will be station assembly, module removal, OMV/OTV berthing and deployment, as an aid to OMV, OTV, and satellite servicing, and, possibly, as an aid to orbiter/station berthing.

A. Space Station Schedule

The space station program begins in April 1985 with an 18-24 month definition-and-preliminary-design study. The design-and-development phase starts in FY 1987 and continues for about a two-year period. The first launch will take place about three years later, in 1992. In addition to the space station itself, there will be an orbital maneuvering vehicle, an orbital transfer vehicle, free flyers, and a space platform. Figure 1 provides a good indication of the many spacecraft that will interact with the space station.

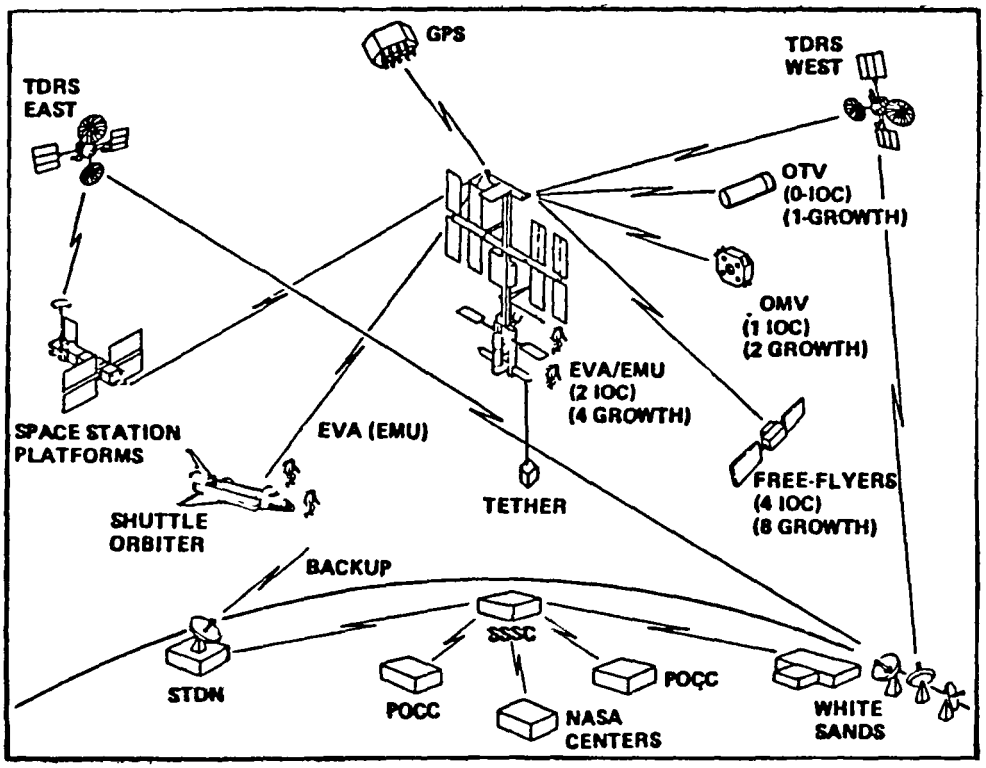


Figure 1: Spacecraft Interacting with the Space Station, (from TRW Space Station Data System (SSDS) Briefing Charts)

To be included in the IOC, the appropriate technology must be available by approximately 1986. A Space Station Advanced-Development Program has been established by NASA for the purpose of adapting generic technologies to space station application requirements, building and integrating prototype components into subsystems for demonstrations in ground-based testbed facilities, and conducting flight experiments, using the space shuttle as necessary. As far as the SSAS is concerned, the programs of interest are

- The manned-systems program, which includes a man-machine interface for teleoperation.

- Attitude control and stabilization, which include autonomous, teleoperator, and robotic control.

- Data management, which includes research and development at various NASA centers: ARC (analysis and simulation), GSFC (end-to-end design), JPL (autonomy and automation), KSC (system checkout), LaRC (technology development), MSFC (systems integration), NSTL (user requirements), and JSC (space station data system and overall testbed management).

B. The Advanced-Technology Automation Committee

Congress has requested that NASA establish an Advanced Technology Automation Committee (ATAC) to report to Congress on April 1, 1985. ATAC was to identify promising automation and robotic technology for the space station, and make recommendations that would comprise an integral part of the definition-and-preliminary design contract for the space station.

The purpose of the Space Station Automation Study (SSAS) was to develop informed technical guidance for NASA in the use of autonomy and autonomous systems to implement space station functions. As defined in a recent report [1],

> "*Autonomy is an attribute of a system/subsystem that will allow it to operate within its specified performance requirements without external intervention for a specified period of time.*"

This definition does not exclude either man or machine from the system/subsystem design. Automation refers to the use of machines to effect control of system/subsystem processes in a predefined or modeled set of circumstances. Automation is a vital tool for implementing an autonomous system.

Good statements of space station automation goals are given as reflecting the automation/autonomy requirements and architectural guidelines of the NASA

Autonomy Working Group (AWG) and the NASA Space Station Concept Development Group (CDG) (Table 1 and Table 2).

The SSAS was conducted by a concepts design team and a technology team. Each member of the concept design team examined particular topics relevant to the space station to determine how the required functions could be automated. The members of that team and the topics they reported on were (1) TRW (satellite servicing), (2) GE (space manufacturing), (3) Hughes (subsystem autonomy), (4) Martin-Marietta (autonomous systems and assembly), and (5) Boeing (man-machine interface). The role of SRI, as the technology team, was to utilize the automation concepts postulated by the first four concept teams to determine what research and development would be required in artificial intelligence (AI) to attain the capabilities implied by these concepts.

C. Previous Studies

There have been several recent studies that are relevant to space station automation. The 1979 NASA Study Group produced the "Sagan report" [4], which concluded that

> "*The overall importance of machine intelligence and robotics for NASA has not been widely appreciated within the agency.*"

and that

> "*The advances and developments in machine intelligence and robotics needed to make future space missions economical and feasible will not happen without a major long-term commitment and centralized, coordinated support.*"

The 1980 NASA/ASEE summer study also concluded that *advanced machine intelligence and automation technology is believed to be essential in evolving toward a major space program capability.* A 1983 report by SRI International recommended the establishment of an AI group within NASA [5]. The 1983 NASA/ASEE summer study [6] recommended that

> "... *major effort and funding should go into the development of manned extra-vehicular activity (EVA*, teleoperation/telepresence, and robot systems.)"

A detailed technical study was carried out by MIT in 1982 for the Marshall Space Flight Center(MSFC). "Space Applications of Automation, Robotics, and Machine Intelligence Systems (ARAMIS)" explored the potential applications of ARAMIS to space activities. The ARAMIS options defined by the MIT study group span the range from fully human to fully machine, including a number of intermediate options. In 1983, the Phase II ARAMIS study dealt with telepresence as applied to five space projects. The gist of the executive summary (Volume 3) is

Table 1: Autonomy/Automation Philosophy prepared by the NASA Autonomy Working Group, of the Space Station Concept Development Group [2].

- Subsystem/system monitoring and control will be performed on board.
- System monitoring and control will be automated.
- Fault detection and isolation will be an automated function for all subsystems.
- Redundancy management, including reconfiguration, will be performed automatically on board.
- Reverification of systems/subsystems elements will be performed automatically on board.
- Near-term (next one to three days) planning and scheduling of operations will be performed on board.
- The degree of automation will increase as the space station matures and new technologies become available.
- Collection and analysis of trend data will be automated on board.
- The space station platform will have at least the same degree of onboard automation as the manned base.

Table 2: Architectural Guidelines prepared by the NASA Space Station Concept Development Development Group [3].

- Automated fault detection, isolation, and recovery will be carried out, giving highest priority to crew life support and primary mission objectives.
- Automated system architecture is distributed hierarchically.
- Fault detection, isolation, and recovery are accomplished at as low a level as possible in the hierarchy.
- The required fault tolerance capabilities may be attained by using either fault-tolerant computers or appropriate network techniques, or both.
- Architecture must facilitate development and test of individual subsystems independently of other subsystems.
- Architecture should minimize subsystem interactions at all levels of architecture. Where interaction is required, it will be performed at the highest feasible level.
- Only processed results will routinely progress upward through the hierarchy. Lower-level data will be accessible at higher levels when needed.
- Architecture will allow manual intervention in all automated processes. Appropriate safeguards should be provided to prevent inadvertent or unauthorized disabling of essential automated processes.

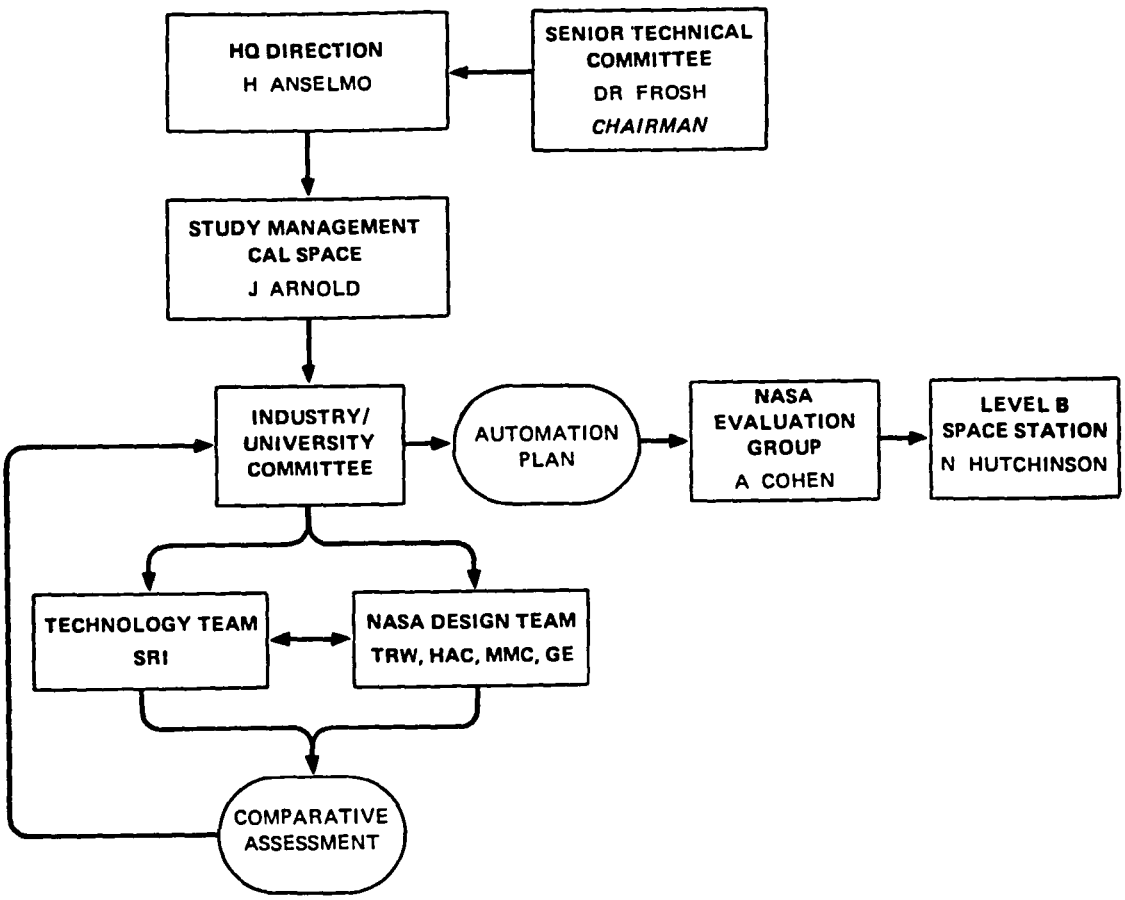


Figure 2: Participants in the Space Station Automation Study

that telepresence is necessary, desirable, feasible, and could be for use in 1990-1992. Advanced telepresence systems will be capable of very complex operations and high levels of autonomy, and a research-and-development program should begin immediately. Volume 1 provides an overview of existing telepresence and an outline of NASA's plans. Volume 2 examines several space projects in detail to see what would be required of a telepresence system. Some of the ARAMIS results are discussed in the chapter on teleoperation.

Most recently, McDonnell Douglas studied the human role in space (THURIS) for MSFC [7]. This one-year effort investigated the role and the required degree of direct human involvement in future space missions.

D. Goals and Methodology of the SRI Study

The goals of the SRI study were (1) to provide guidance with respect to the state of the art in artificial-intelligence (AI)-based technologies; (2) review the results of the concepts design contractors to determine the AI capabilities required by the designs; (3) delineate a series of demonstrations that would indicate the existence of these capabilities; and (4) develop a research-and-development plan leading to such demonstrations. As a separate issue, advanced techniques for the space station's data management system were also to be investigated.

The methodology used in the SRI study consists of the following steps:

(1) Examine automation concepts prepared by the concepts design contractors and determine needed automation capabilities.

(2) Derive sequences of demonstrations leading to the desired automation capabilities.

(3) Derive research and development plans leading to technology for carrying out these demonstrations.

A similar focus on demonstrations is used in the DARPA Strategic Computing Plan. Basing the R&D plan on a sequence of postulated demonstrations has several advantages. For example, it provides verifiable milestones of increasing difficulty. It establishes a tangible goal for AI-based R&D and makes it possible to ascertain when the results of such research are actually needed

We first reviewed the material provided by the concepts design contractors and identified the implied automation capabilities required. After determining the latter, we then postulated a series of demonstrations that would verify the existence and motivation of these capabilities. Finally, for each of the AI-based technologies, the relevant research and development to carry out the demonstrations were indicated. Chapter 2 discusses the results of the concept design contractors, while Chapter 3 identifies the applications requiring AI and briefly discusses this technology. The balance of this report deals with demonstrations and the associated R&D required. Each topic is described briefly, its relevance to the space station indicated, and the state of the art of each discussed. Finally, the demonstrations and the research and development for each topic are described in greater detail.

REFERENCES

1. "Autonomous Systems: Architecture and Technology," (Tech. Rep. JPL D-1197), Pasadena, California, Jet Propulsion Laboratories, February 1984. Prepared for the Office of Aeronautics and Space Technology, National Aeronautics and Space Administration.

2. W. Holmes, "Autonomy, Automation, Robotics," (Tech. Rep.), Washington, D.C., NASA Headquarters, December 1983. Autonomy Working Group Presentation to Space Station Concept Development Group.

3. R. Staehle, "Extent of Automation of the Space Station from an Operational Viewpoint," (White Paper), Pasadena, California, Jet Propulsion Laboratories, August 1983. Space Station Program Description Document, book #6, Appendix B, 2nd-Level White Paper.

4. C. Sagan, et al., "Machine Intelligence and Robotics: Report of the NASA Study Group," (Tech. Rep. N80-30086), Pasadena, California, Jet Propulsion Laboratories, September 1979.

5. D.R. Brown and P.C. Cheeseman, "Recommendations for NASA Research and Development in Artificial Intelligence," (Final Report), Menlo Park, California, SRI International, April 1983. SRI Project 4716.

6. D. Bershader and L. Leifer, "Autonomy and the Human Element in Space," (Tech. Rep.), Stanford, California, Stanford University, December 1983.

7. "The Human Role in Space (THURIS)," (Final Briefing NASA 8-35611), Huntington Beach, California, McDonnell-Douglas Astronautics Company Co., September 1984.

II CONCEPT DESIGNS

This chapter summarizes the Concept Designs of the Space Station Automation Study (SSAS) contractors: General Electric, TRW, Martin-Marietta, Hughes, and Boeing. We describe the missions used by the contractors, the activities that occur during each mission, and the chronology of the mission.

A. Space Manufacturing

The General Electric Company studied the automation technology required for manufacturing operations, with emphasis on the production of gallium arsenide crystals and wafers, and of gallium arsenide VLSI microelectronic chips, [1]. The manufacturing-facility design concept shown in Figure 3 depicts an automated production facility for GaAs crystals and wafers and a processing facility for microelectronic chips.

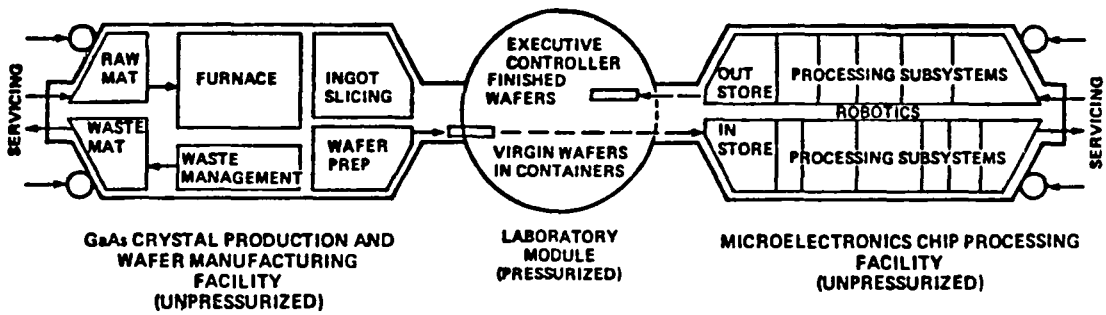


Figure 3: GE Design Concept for a GaAs Manufacturing Facility

The automation aspects can be summarized as follows:

Crystal Production and Wafer Manufacturing. The crystal and wafer production facility postulated by GE is shown in Figure 4. Much of the automation is in the form of process mechanization schemes similar to those used in factories today for materials handling and manufacturing. However, primarily because of their flexibility, robots are conceptualized for materials handling, servicing, and maintenance functions associated with the furnace. Their operating profiles and schedules can easily be altered or upgraded by software, and the

configuration of end effectors and their functions can easily be changed for the required application. AI systems will play an important role in operation of the facility, primarily in process control, troubleshooting, and maintenance. AI applications are also contemplated for the kind of situation in which space station power has been disrupted at some point partway through a crystal growth cycle.

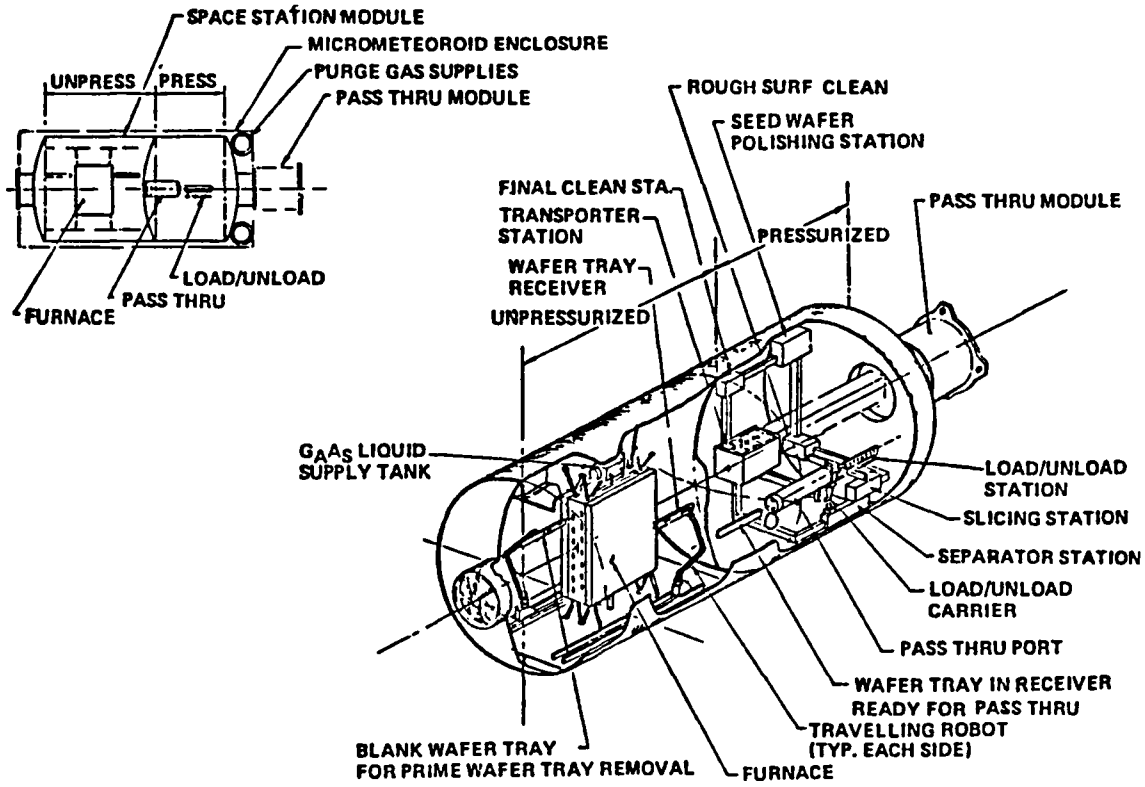


Figure 4: Crystal Production and Wafer Manufacturing Facility

Chip-Processing Facility. The chip-processing facility is shown in Figure 5. The automated handling of wafers requires versatile robotics and complex scheduling to process many wafers at a time efficiently. Robotics is used to transfer wafers in containers from storage through each subsystem and in the desired sequence. Control is highly automated, but is supervised and monitored on the ground and in a pressurized laboratory module by a crewman. Room is provided for access by the crew if for purposes of repair and maintenance. Eventually nearly all repairs, servicing, and maintenance would be done in the spaceborne facility through automation. AI techniques will be used to identify any abnormalities in the production process.

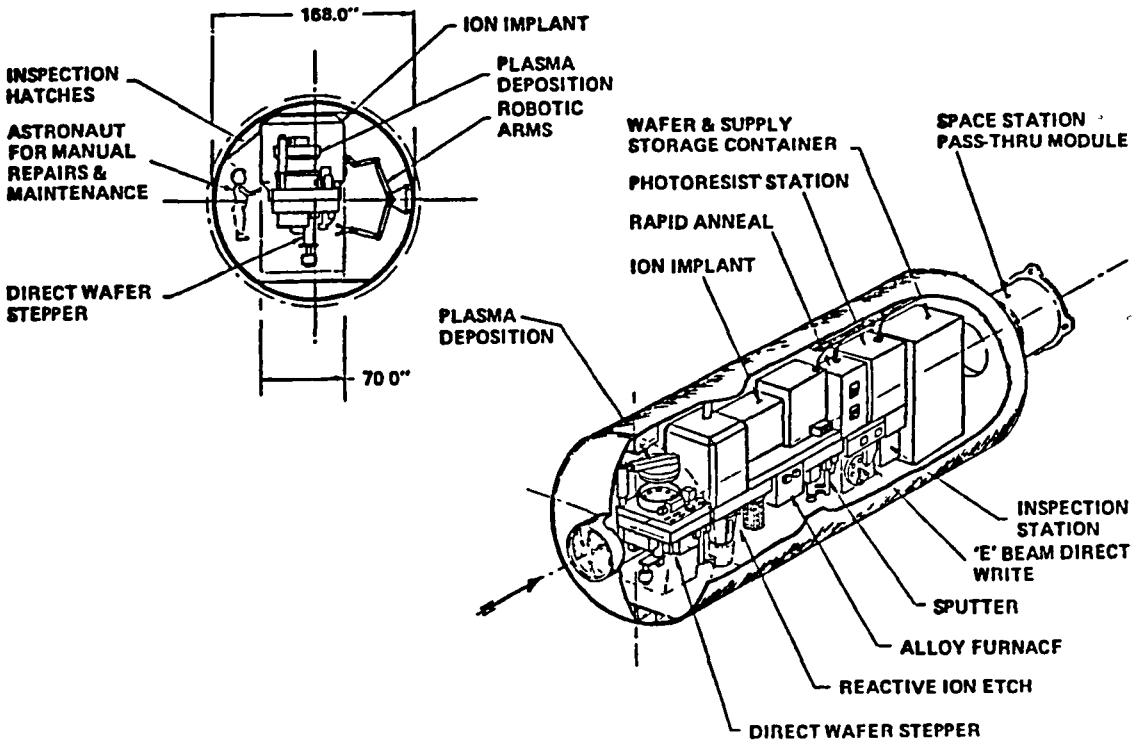


Figure 5: Microelectronic Chip Processing Facility

The GE report describes the movement of materials through the facility, the operations that must be performed on them, and the problems encountered in the automatic supervision, maintenance and repair of this type of facility. GE concluded that, while very versatile industrial robots are in extensive use today, those conceptualized for space applications will be of a very different design. *They must be able to operate in a hostile environment of hard vacuum with potentially high thermal gradients and radiation. While microgravity allows their design to be lightweight, different kinematics and dynamics will exist. Different approaches to actuators and end effectors must therefore be developed. Gravity can no longer be used as a helper to catch things or hold them in place.* GE indicated that the more autonomy is developed, the more reliable, serviceable, and easily repairable must be the equipment. *It will be difficult to provide the space station crew with the kind of access, information, and resources needed to adjust or repair highly automated systems in the confines of a space facility to the degree possible in an earth-based factory.* GE feels that the major challenge of space manufacturing is maintenance and repair. Without the automation

capabilities to accomplish these functions, manufacturing in space will be unattainable.

GE stressed the need for AI-based technology for manufacturing facilities that will provide efficient control of troubleshooting, maintenance, and options for corrective action. *Development of expert systems to perform the job even better must await expertise to be gained in operating the manufacturing system during development and in space. As experience is gained, more hardware and software automation can be accomplished, thus making space factories more productive...*

GE recommended five specific research-and-development programs for establishing a space manufacturing facility: (1) a space manufacturing concepts development study, (2) a space robotics system experiment, (3) a materials management study, (4) materials-handling experiments, and (5) a space manufacturing AI applications study dealing with expert system control, maintenance, and troubleshooting. The GE schedule for these recommended space station automation development studies is shown in Figure 5. Basically, automation of the maintenance of manufacturing facilities will require significant development, since an automated device must be capable of handling moderately complex repair, cleaning, and refurbishment tasks in a microgravity environment. In addition, expert systems for process control and maintenance are required to monitor the processing and to analyze problems that arise.

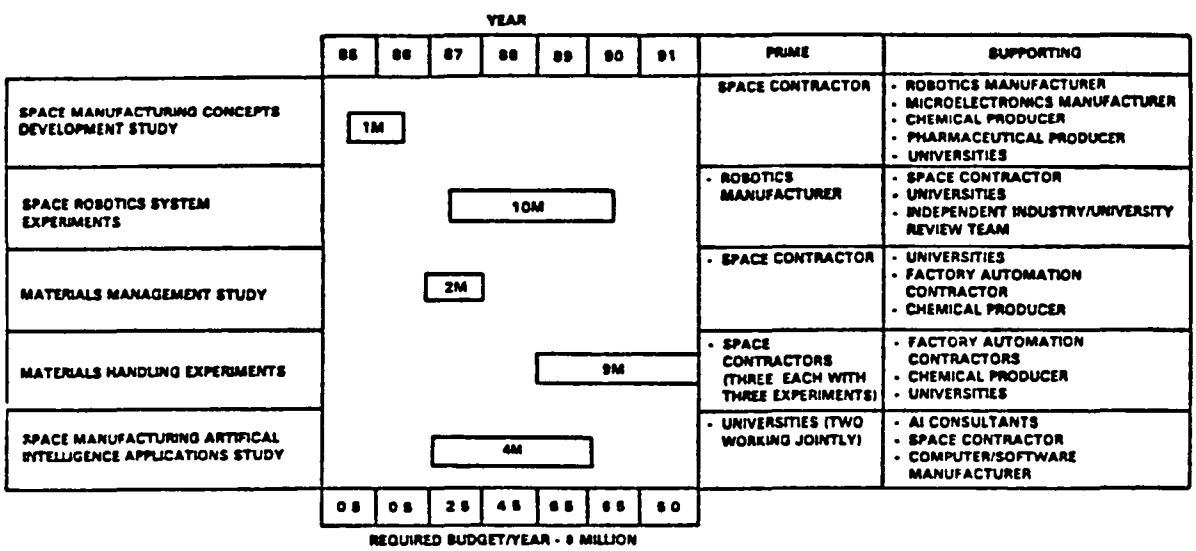


Figure 6: Summary of GE Automation Requirements for Chip Processing

B. In-Orbit Maintenance

Typical satellite-servicing functions are deployment, inspection, maintenance, resupply, repair, modification, cleaning or resurfacing, and replacement. The four automated servicing missions used by TRW in their portion of the study, shown in Figure 7, were as follows:

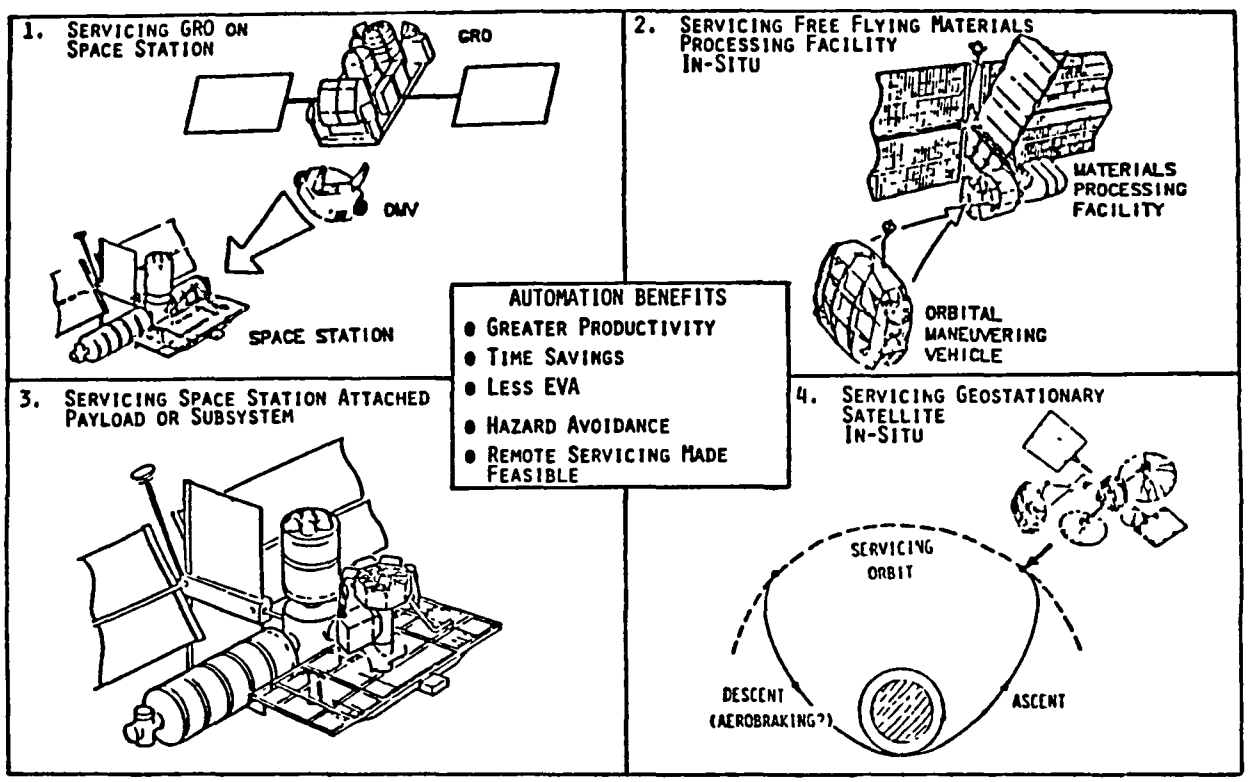


Figure 7: Automated Servicing Reference Missions Considered by TRW

(1) Servicing of a low earth orbit (LEO) satellite, e.g., the Gamma-Ray Observatory (GRO), at the space station, with orbit transfer by an orbital maneuvering vehicle (OMV). The OMV retrieves the GRO from 400-km-high orbit, and berths at the space station. A comprehensive series of status tests is carried out on the GRO. Failed units are replaced, their propellant refilled, and the GRO is checked out and redeployed. The automation requirements include remote control of GRO retrieval, automated rendezvous and docking at the space station, load handling and transfer by teleoperation, propellant

refill, and automated tests and checkout.

(2) Servicing free-flying, co-orbiting materials-processing facility (MPF) *in situ*, which consists primarily of periodic resupply and the harvesting of finished products. Here the OMV is attached to a servicing module carrying fresh sample material. The OMV transfers to and performs rendezvous and berthing at the MPF, and then exchanges sample magazines at the MPS under remote control. The OMV performs orbit reboost for the MPF, returns to the space station, delivers the finished samples, and is refurbished for the next use. The automation requirements include load handling and transfer at the space station by teleoperation, rendezvous, docking and berthing at the MPF, sample magazine replacement at the MPF, automated checkout of the MPF, and orbit reboost of the MPF by the OMV.

(3) The repair/refurbishment or changeout of payloads or subsystems attached to the space station includes inspection of the payload or subsystem to be serviced, call for and reception of required parts and supplies via the orbiter, and transfer of the object being serviced to and from the work station. The repair, refurbishment, and module replacement are performed, and the payload or subsystem is then checked out and restored to normal operation. The automation requirements include load handling and transfer, automated tests, diagnostics, checkout, and module replacement by teleoperation.

(4) Servicing of a geostationary satellite *in situ* by using a recoverable orbital transfer vehicle (OTV) to perform the ascent and descent to and from synchronous orbit, carrying supplies, replacement parts, tools, and support equipment such as a remote/robotic servicer. The servicing module is attached to the OTV, transferred to synchronous orbit, and docked with the target satellite. The satellite is checked out and the failed module replaced, The satellite is refueled, as appropriate, and the OTV returns to the space station. The automation requirements are load handling and transfer on the space station, and assembly of the servicing vehicle with the OTV. The orbit transfer, rendezvous, and docking are automated, as are the inspection, module replacement, and refueling.

TRW states [2]: *In missions 1,2, and 4, a remotely controlled orbital*

maneuvering or transfer vehicle (OMV,OTV) with a "smart front end" is utilized to perform propulsive functions and repair or resupply services.. All of the three principal automation fields, teleoperation, robotics, and AI, are utilized in the servicing missions investigated. However, teleoperation with the human in the loop is used prominently, whereas fully robotic systems will be employed less frequently due to the diversity and unpredictability of the servicing tasks to be performed.

Telepresence was the principal automation discipline required for servicing, with human operator involvement to handle task diversity and unforeseen situations. The GEO satellite servicing demands more reliance on full robotic mode, using only supervisory control, with teleoperation serving as a backup. Massive support by the space station data system is necessary—in the planning, sequencing, and execution of tasks, and in assisting the crew in maintenance procedures.

Some of the manipulator concepts for maintenance applications are shown in Figures 8 to 11. Note that these include the following:

- Dexterous manipulators that are attached to a structure. A manipulator attached to a telescoping-base pedestal, and one on a crawler are shown.

- A pressurized mobile work station concept; this can be an enclosed cherry picker, a manned free flyer, or a railed work station. This mobile work station permits close teleoperation, with less fatigue for the crew than in EVA.

- A teleoperated and/or autonomous servicer with two manipulators and stereo vision.

C. Assembly of Large Space Structures

Martin-Marietta examined automation of the assembly of large space structures, [3]. The company's mission models are as follows:

- 1991: Assemble IOC Space Station. The assembly is carried out over a series of seven shuttle flights. A mobile remote manipulator system (MRMS), a 21-foot extension of the standard RMS, figures strongly in this assembly. Construction begins with the first flight; an initial structure is erected, the rails, boom, and arms are unfolded, and the MRMS is set up. Subsequent flights involve removal of packages from the payload bay, transporting them, and attaching them to the structure.

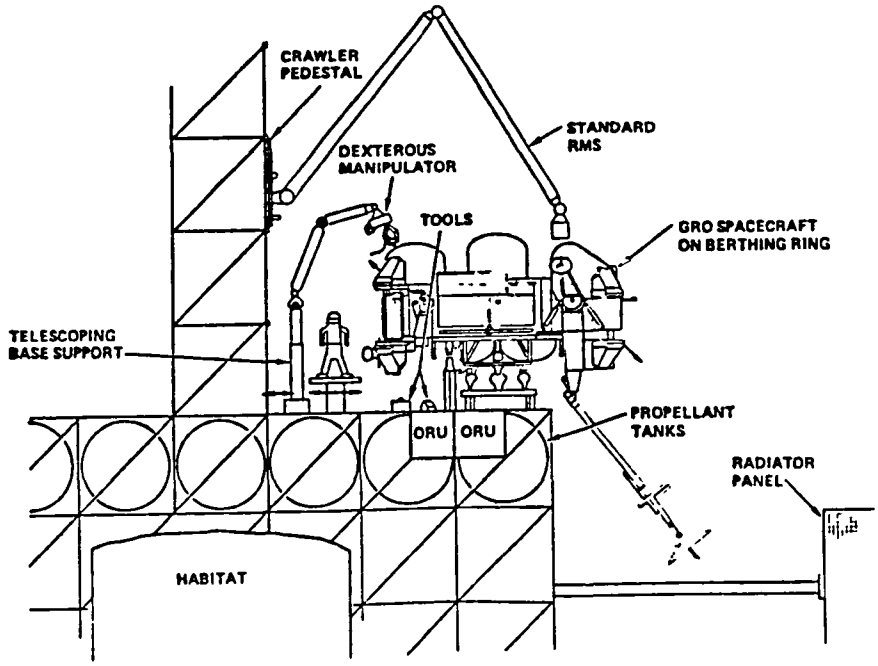


Figure 8: Base-Supported Dexterous Manipulator

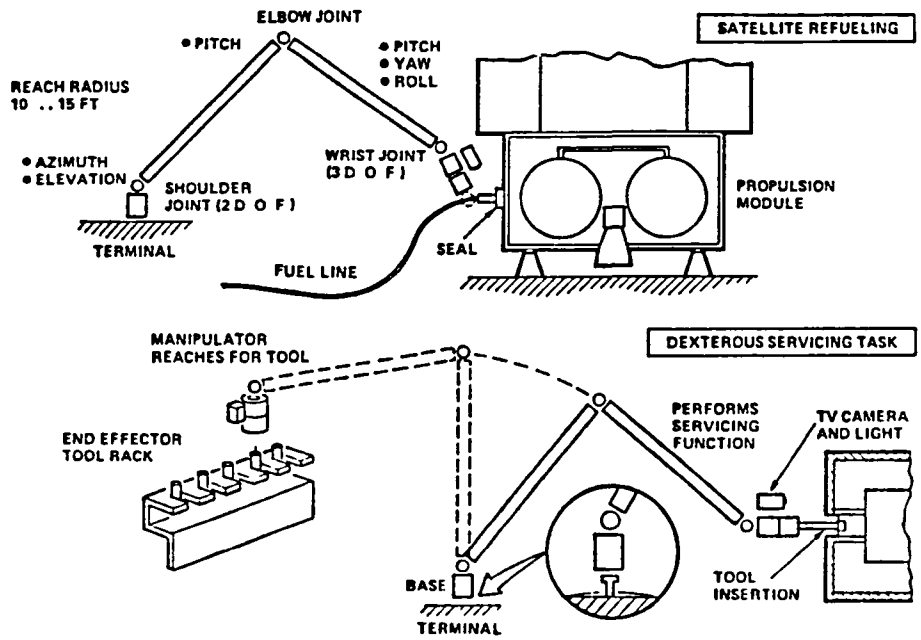
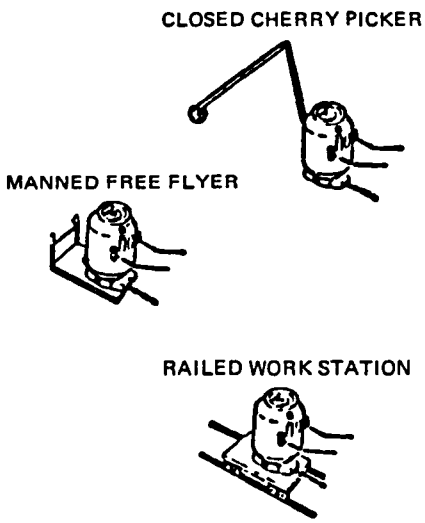
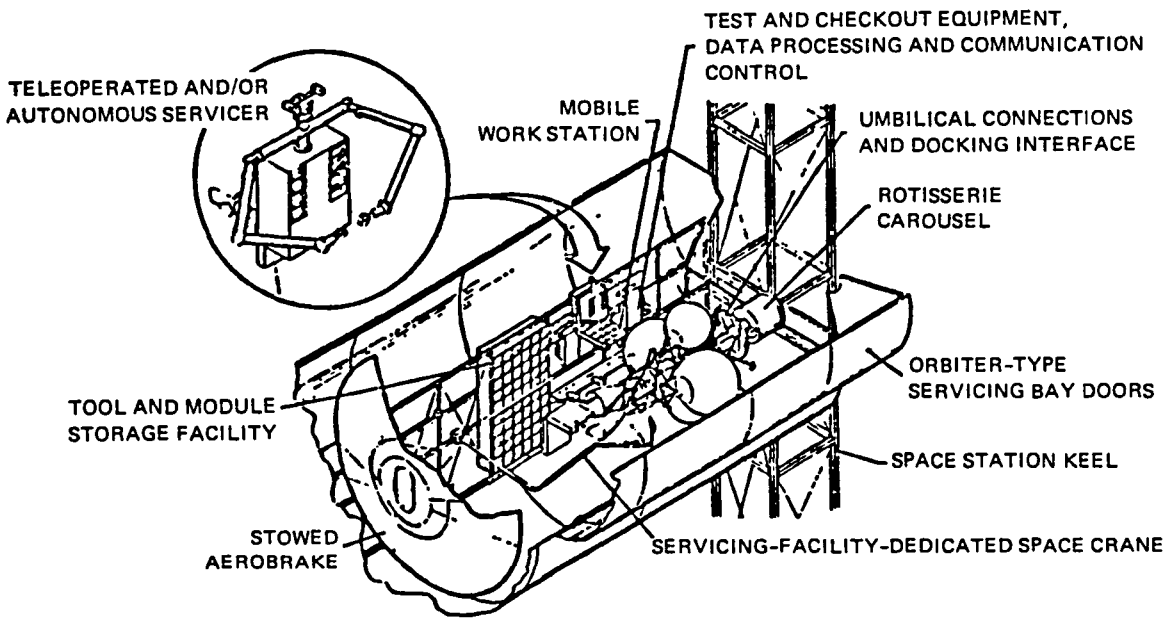


Figure 9: Portable Dexterous Manipulator Concept (TRW)



- ENCLOSED CHERRY PICKER
- INITIAL DESIGN STUDIES BY GRUMMAN AND OTHERS IN LATE 70's, EARLY 80's
- HYBRID EVA/IVA CONCEPT
- IN EARLY SPACE STATION PHASE SUPPORTS SERVICING WITH EMPHASIS ON DIRECT CREW INVOLVEMENT AND PROVIDES ADDED TELEOPERATION CAPABILITY
- FEATURES
  - DIRECT VISUAL INSPECTION
  - CLOSE-RANGE TELEOPERATION
  - EXTENDED CREW ENGAGEMENT POSSIBLE (MORE THAN SIX HOURS OF TYPICAL EVA DURATION)
  - LESS FATIGUE
  - LESS CREW EXPOSURE TO POTENTIAL EVA HAZARDS
  - SAFETY ISSUES INVOLVE RMS TRANSFER
- DECENTRALIZED-CONTROL SUBSTATION
- FLEXIBLE DEPLOYMENT
- FULLEST ACCESS TO DATA SYSTEM SUPPORT
- ACCESS TO TETHERED SATELLITES
  - IN FREE-FLYING MODE
  - BY MOVING ALONG CABLE

Figure 10: Pressurized Mobile Work Station (TRW)



SOURCE Martin-Marietta Aerospace "Technology Development Missions for Early Space Station Satellite Servicing", Final Review, October 9, 1984

Figure 11: Teleoperated and Autonomous Servicer (Martin-Marietta)

- 1992-1994: Expand Space Station. The space station is expanded by adding a satellite-servicing and an OTV hanger-and-servicing facility.

- 1997: Assemble Large Spacecraft. A large deployable refector (LDR) is assembled at the space station. The LDR structural elements and reflector segments are delivered to the space station in two orbiter missions. The LDR is assembled on the service structure strongback, using the manned maneuvering unit (MMU) and the space station RMS/work platform. The LDR is deployed into operational orbit by the OMV, which is returned to the space station and refurbished.

- 2000: Assemble Geostationary Platforms. Advanced large commercial communications system (LM-7) and manned geostationary platform (LM-13) are assembled.

Some of these missions are shown in Figures 12 and 13.

Martin-Marietta considered the following assembly methods: EVA, remote manipulator system (RMS), the mobile RMS (MRMS), an "automated" system, and EVA using a foot restrainer at the end of the RMS (MFR).

The Martin-Marietta analysis of various operations for space station assembly on each flight of the shuttle is shown in Figure 14. Note that both EVA and MRMS are used for a high percentage of the time.

D. Automation of Subsystems and Mission Ground Support

Hughes Aircraft was assigned the task of developing an automation concept for the autonomous operation of space system subsystems. The objective was to identify those functions associated with the operation of such subsystems as electric power, thermal control, communications, and tracking. System monitoring and control were also included as a study task. As indicated in their final report:

Hughes developed a concept with unconstrained automation for the operations of the space station. In that concept, the station has an automated system for system monitoring and control that detects, isolates, and recovers from failures. Crew members are thus freed from routine monitoring and sequencing through malfunction procedures and can then devote most of their working hours in support of payload operations....Users have a high degree of flexibility in the operations of their payload. They are able to directly command

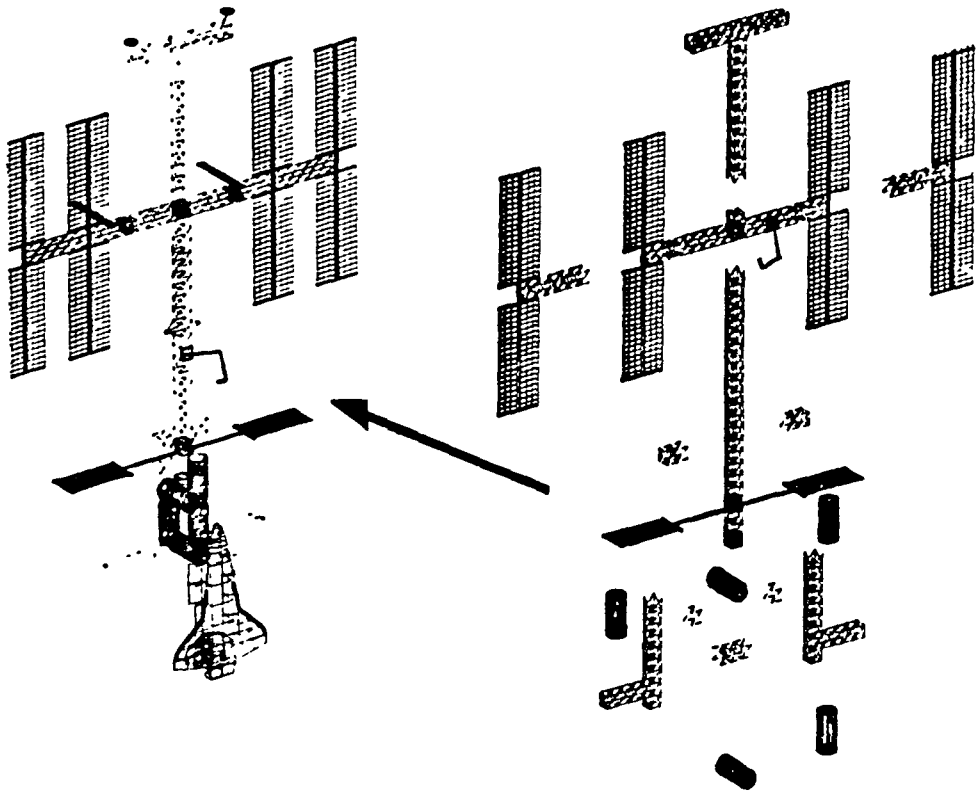


Figure 12: Space Station Assembly

- DELIVER LARGE DEPLOYABLE REFLECTOR (LDR) STRUCTURAL ELEMENTS AND REFLECTOR SEGMENTS TO SPACE STATION IN TWO ORBITER MISSIONS
- ASSEMBLE LDR ON SERVICE STRUCTURE STRONGBACK USING MMU AND STATION RMS/WORK PLATFORM
- DEPLOY LDR TO OPERATIONAL ORBIT WITH OMV
  - RETURN OMV TO SPACE STATION AND REFURBISH

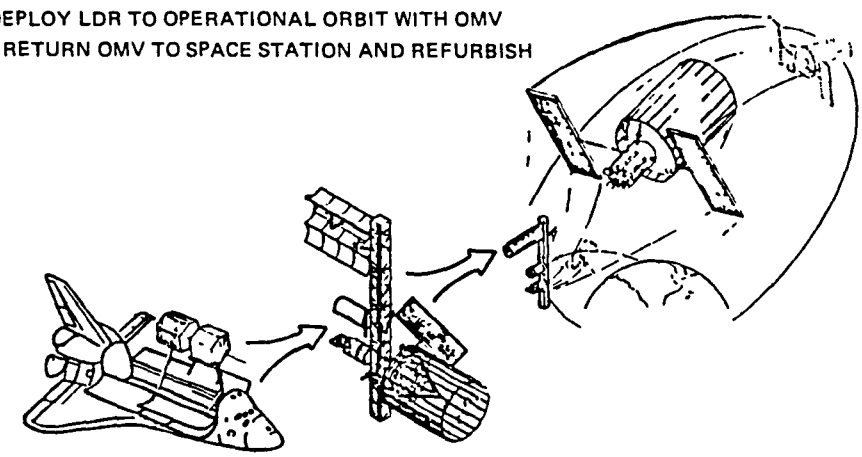


Figure 13: Assembly of Large Spacecraft

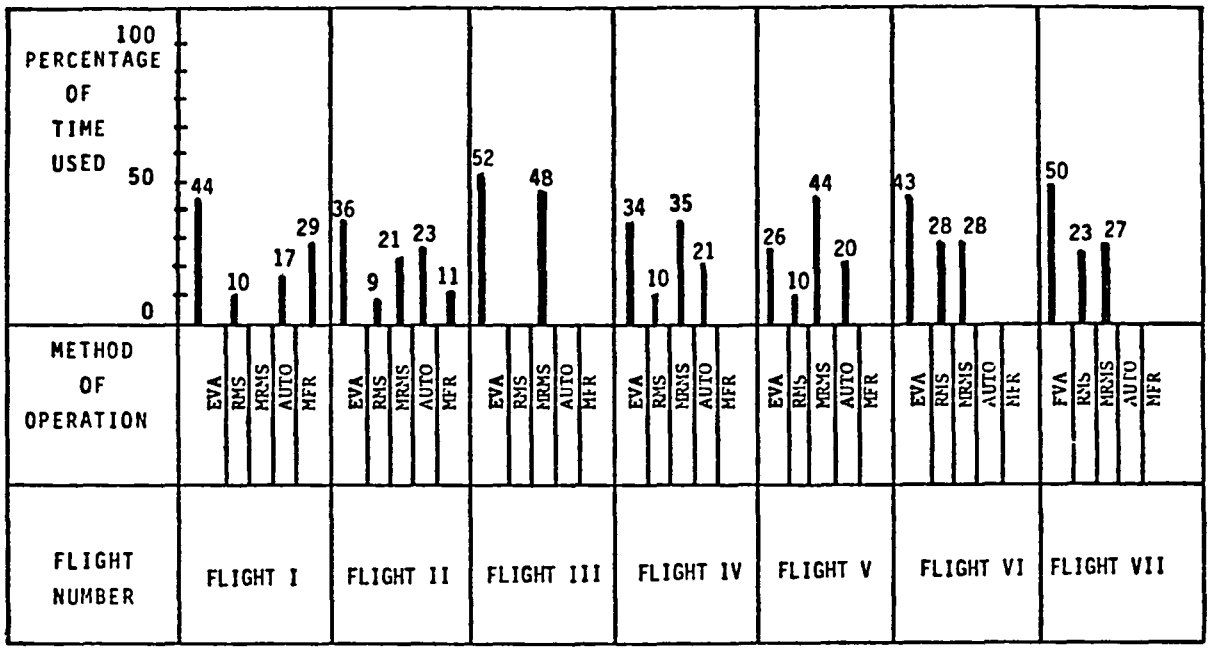


Figure 14: Projected Percentage of Operational Methods During Each Flight (Martin-Marietta)

their payload without coordination or scheduling by a ground mission control group ..

The Hughes report [4] examines key functions in the areas of communications, power and thermal management, and system monitoring and control. The AI-related capabilities required were speech input/output for crew command and control, and expert systems for scheduling of communication services and coordination of electric power and thermal resources.

E. Operator-Systems Interface

The Boeing Aerospace Company studied the Operator-System Interface (OSI) for an extra-vehicular (EV) robot system that carries out space station maintenance tasks. OSI scenarios for that system were developed, and the associated technologies were assessed. Boeing concluded [5] that a rudimentary experimental EV robot and OSI system with a limited supervisory capability could be produced by mid-1990, and that it is technically feasible to develop an automated OSI capable of supervisory management of an EV robot by the year 2010. However, significant advances in natural language understanding and in

automated reasoning are required to attain an advanced OSI. To that end, Boeing described an R&D program that could be initiated by NASA.

F. Timing of Concepts

The missions postulated by the concepts design team were consolidated as shown in the chronology of Figure 15. Note that the missions cluster around the IOC period, and that the latest one is scheduled for the year 2000. In the more detailed analysis given by us in subsequent chapters, we may find that one or more of these milestones cannot be met because the necessary capability cannot be developed in time.

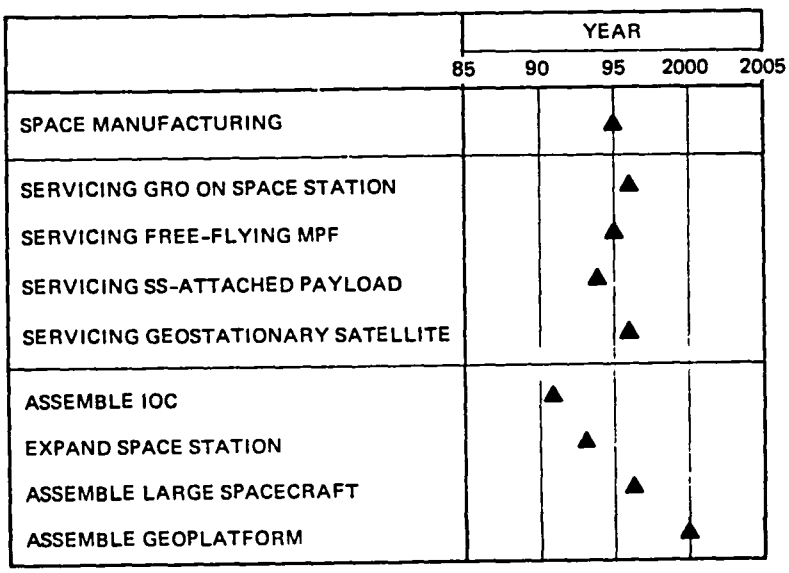


Figure 15: Chronology of the Missions Considered by Contractors

This chapter has summarized the contractor's concept designs. The next chapter discusses the automation capabilities implied by these concepts.

REFERENCES

1. "SSAS: Automation Requirements Derived from Space Manufacturing Concepts," (Final Report NR 740-9), Valley Forge, PA, General Electric Space Systems Division, November 1984.

2. "SSAS: Satellite Servicing," (Final Report Z 410.1-84-160), Redondo Beach, California, TRW Space and Technology Group, November 1984. NASA Contract NAS 8-35081.

3. "SSAS: Autonomous Systems and Assembly," (Final Report MCR84-1878), Denver, CO, Martin Marietta Aerospace, November 1984.

4. "SSAS: Automation Study for Space Station Subsystems and Mission Ground Support," (Final Report F5713), Los Angeles, California, Hughes Aircraft Company, November 1984.

5. "SSAS: Operator System Interface," (Final Report), Seattle, WA, Boeing Aerospace, November 1984. Concept Definition and Technology Assessment.

III. ANALYSIS OF AI TECHNOLOGY NEEDS

III ANALYSIS OF AI TECHNOLOGY NEEDS

The purpose of this chapter is to identify those applications that require artificial intelligence, to indicate when they can be implemented, and then to describe the specific AI technologies needed for such applications. Subsequent chapters will discuss these technologies in detail.

A. Major Applications of Automation

To determine what AI-based technology will actually be needed, we examined the SSAS contractor and other reports dealing with space station automation. From our analysis of these reports, it became evident to us that AI technology will be essential for the following applications, with the first two of primary importance:

(1) Satellite servicing. Capability of servicing satellites at the space station and *in situ* can result in large savings and increased scientific return based on extended life for many missions, such as the GRO, the Space Telescope, and the Space Infrared Telescope Facility. In addition, many technical disciplines that can be transferred to industrial automation are affected.

(2) System monitoring and diagnosis. The use of systems for monitoring and diagnosis will be required for the space station because of the complexity and evolution of the system. Such systems relieve the crew of more routine duties and would reduce the need for ground operations support in this area. The technology developed would contribute to industrial applications on earth.

(3) Space Manufacturing. There will be a need for teleoperation/robotics concerned with maintaining space manufacturing equipment. In addition, expert systems capable of supervising quality control, operating the production system, and maintaining the equipment will be necessary.

(4) Assembly of Space Structures. Teleoperation will play a major role in unloading and moving structural elements from the space shuttle; it will also assist in assembling these structures.

The schedule anticipated for these applications is shown in Figure 16. It will be noted that assembly of the space station is planned for in the early 1990s, system monitoring can start at IOC, and most of the manufacturing and servicing applications are initiated about midway through this last decade of the century.

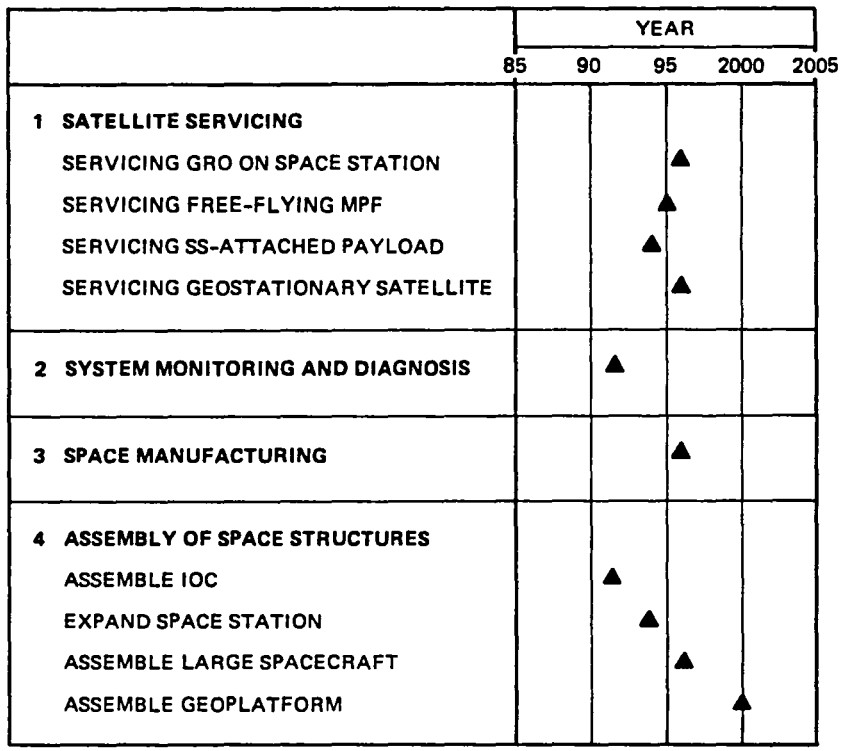


Figure 16: Implementation Schedule for the Four Application Areas

We discuss the four application areas below, indicating the automation technologies required for each.

1. *Satellite Servicing*

The level of servicing capabilities required will depend strongly on the design of the equipment being serviced. Affected by this design factor is the ease of replacing of modules, of connecting and disconnecting cables and fuel lines, and of automated docking. Diagnosis requires built-in test equipment and the availability of testing points. Chapter 4 deals with these and other questions related to design.

TRW identified the servicing functions required for its reference missions as (1) ORV replacement—at the SS and *in situ*, (2) payload changeout—at the SS and *in situ*, (3) refueling—at the SS and *in situ*, (4) mating of the OMV or OTV

with the payload, (5) GEO service (all functions), and (6) MPS resupply and harvesting *in situ*. The servicing functions at the space station utilize some augmented form of the RMS and can always resort to EVA when difficulties are encountered, but *in situ* maintenance must do without that luxury. Even if a pressurized teleoperation pod is used, its manipulators must be capable of fine manipulation. The most demanding capability is required of an OMV working at a GEO orbit. Because the earth's geostationary altitude lies in the very midst of the Van Allen radiation belt, any manned station there would have to carry radiation shielding to protect the crew. Consequently, the motivation for more advanced teleoperation/robotic capabilities will be the importance of *in situ* servicing, particularly GEO servicing.

The key automation technology identified by TRW for each of these capabilities is listed in Figure 17. Note that dexterous manipulation figures very strongly, as does automated test equipment.

SERVICING FUNCTION	BENEFIT	BENEFIT CATEGORY	KEY AUTOMATION TECHNOLOGY DM	RV	ES	AT	AR	LT	COST CATE-GORY	COST - BENEFIT INDEX
1. ORV REPLACEMENT										
- AT SS	ESSENTIAL S/C SERVICE FUNCTION	2	•			•			1	4
- IN SITU		3	•	•		•	•		2	4
2. P/L CHANGEOUT										
- AT SS	ENHANCES S/C UTILITY	2	•			•			2	4
- IN SITU		3	•	•	•	•	•		3	4
3. REFUELING										
- AT SS	ESSENTIAL SERV. FUNCTION FOR MOST S/C	2	o			•		•	1	6
- IN SITU		3	o	•		•	•	•	3	4
4. MATE OMV, OTV TO PAYLOAD	REDUCES EVA REQUIREMENTS	1	•	•		•			2	2
5. GEO SERVICE (ALL FUNCTIONS)	ESSENTIAL LONG TERM GOAL	3	•	•	•	•	•	•	3	4
6. MPS RESUPPLY & HARVESTING IN SITU	ESSENTIAL TO COMMERCIAL MPS PROGRAM	3	•			•	•	•	2	5

LEGEND: DM - DEXTEROUS MANIPULATOR ES - EXPERT SYSTEM AR - AUTOMATED RENDEZVOUS
RV - ROBOT VISION AT - AUTOMATED TEST EQUIPMENT LT - LIQUID TRANSFER

Figure 17: Key Automation Technology Required for Servicing

The schedule for key technologies needed to achieve these capabilities, as indicated by TRW, is shown in Figure 18. Note their assumption that, with

adequate funding, a dexterous manipulator could be available and a demonstration of satellite servicing carried out prior to IOC.

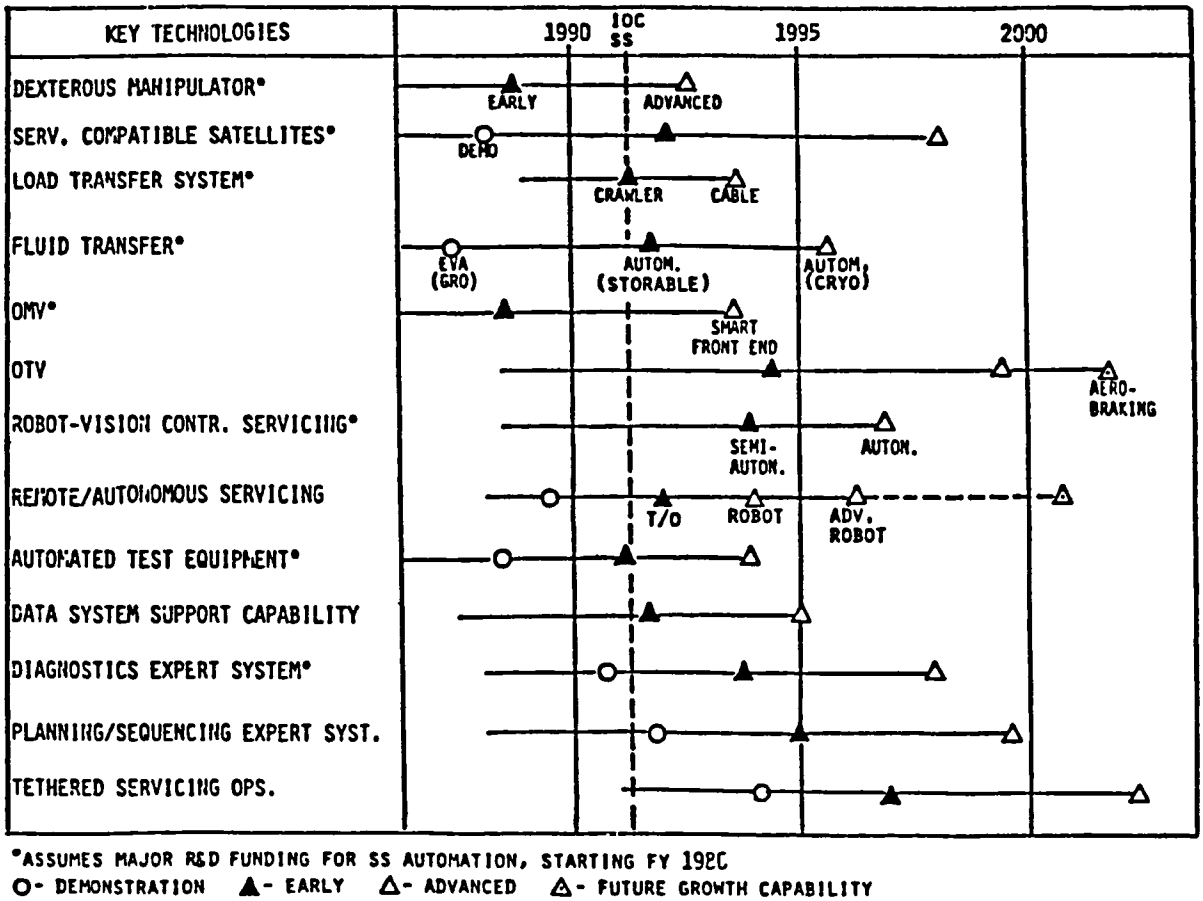


Figure 18: TRW Automated-Servicing Forecast

2. *System Monitoring and Diagnosis*

Widespread use of monitoring and diagnosis systems that incorporate expert knowledge can be expected, beginning at IOC and continuing as the space station evolves Such systems will be used both in an interactive mode, as an aid to the crew, and in an automatic mode. Among prospective applications are the following:

- Diagnosis/repair. Systems capable of maintaining space station equipment by diagnosing problems and suggesting repair procedures. Such systems will be required for the space station

because of the complexity and potential evolution of the system.

- Manufacturing-control systems. Systems capable of quality control, control of production, and maintenance will be required so that manufacturing can proceed without the need for intervention by the crew.

- Control of a space station subsystem. Systems that can augment conventional control systems, e.g., the power control subsystem, to provide expert analysis when problems arise and to suggest courses of action.

- Mission control substitute. Systems that replace various mission control functions by providing onboard capabilities. This will be required to decrease the cost of space station operation.

There will not be enough specialists on board to handle all of the required system maintenance, nor would the availability of such systems would decrease the need for ground operations support in this area. The technology developed would contribute to industrial applications on earth.

An important long-range capability would be the inclusion of enough "commonsense knowledge" about repair processes in general so that an expert system could deal with a variety of diagnostic problems.

3. *Space Manufacturing*

The manipulator concepts developed by GE were described in Chapter 2, in which Figure 4 depicts a traveling robot running along rails on either side of the module. Since the module is a somewhat cluttered environment, one would expect the robot to have difficulty reaching around obstructions to assemble or disassemble portions of the production system. The chip processing facility (Figure 5) shows a two-armed robot on a rail system at one side and access for an astronaut to perform manual repairs and maintenance. Again we have a very cluttered environment that allows little room for manipulations. The capabilities required of teleoperated devices in this application might be as follows:

- The arms could travel on rails, as shown in the GE design.

- Arms might be available on both sides and above the manufacturing space.

- The arms would have to be modular and also repairable by other arms in the manufacturing module.

- The end effectors would be removable and at least some of the arms should be capable of fine manipulation when necessary.

- The module would provide adequate lighting that can be controlled remotely.

GE postulated an expert process controller (EPiC) as follows:

The complexities of chip manufacturing, coupled with the inherent uncertainties associated with space manufacture, give rise to the need for an integrated "expert." EPiC offers a knowledge base from which the space station operator can draw detailed explanations to implement timely process adjustments. The magnitude of the manufacturing process will most likely warrant a dedicated process engineer on board the space station as well as ground support, particularly during startup. The process complexities preclude any one individual from possessing the expertise necessary to detect all implicit processing deficiencies, or correct all deficiencies once they have been identified. The development of EPiC will be an evolutionary process.

4. *Assembling Space Structures*

The level of automation capability required for assembly will depend to a great extent on the cleverness with which the structures and the connecting joints are designed.

The most advanced work in simulating assembly of space structures has been carried out over the past five years by the MIT Space Systems Laboratory; by following their reports [1], one can learn a great deal about the capabilities required. The MIT Beam Assembly Teleoperator (BAT) is a remotely controlled free-flying device capable of assembling structures in neutral buoyancy. An unexpected result of their experiments is that productivity in kilograms per crew-hour does not yet appear to be strongly correlated with the kind of assembly aids used or with the number of people working on the structure at any one time. If this result holds, there will not be any persuasive motivator for teleoperated devices as an aid in assembling space structures. The BAT will be tested further in a fully-developed structural assembly configuration at the NASA Marshall

Space Flight Center this year.*

In the Martin analysis, the mobile remote manipulator system (MRMS) plays an important role in moving, positioning, and joining large structural members. MRMS evolves from a single teleoperated arm to multiple arms, then finally, beyond the year 2000, to "teleautomation". (This sequence is depicted in Figure 18.)

B. Key AI-Based Technologies

The key AI-based technologies required for these applications are teleoperation/robotics and sensors for the manipulation of objects, expert systems to aid in monitoring and diagnostics, and automatic planning to schedule space station resources and determine which actions by autonomous robotic devices are required. We coordinated the scheduling estimates obtained by TRW, Martin, GE, and Grumman [2] in these topic areas, as shown in Figure 19.

1. *Teleoperation and Robotics*

The TRW, Martin, and GE concepts all require similar teleoperation and robotics capabilities—initially under human teleoperation control, then semiautomated control, and finally in an automated mode. Although there is much in common in the three classes of application, each has a different emphasis because of the differences in missions. TRW requires the ability to perform a variety of maintenance functions in both a controlled and uncontrolled environment, using both gross and fine manipulation, at both near and remote distances from the space station. Martin's assembly missions tend to deal with controlled, known situations—generally requiring gross manipulation of large structural members, with the work area usually near the shuttle or space station. GE's mission is concerned with a controlled, but cluttered, "factory-like" environment that often necessitates fine manipulation and must occasionally cope deal with unexpected situations.

As delineated by Martin, the manipulator control system for carrying out the assembly of space structures evolves through the following stages:

---

*The BAT, a self-contained unit with onboard propulsion and power systems, currently carries two manipulators· a dexterous five-degrees-of-freedom arm and a specialized effector. The latter is designed to grasp the 4-inch cylindrical beam and, by means of a small drive wheel in the end effector, move the beam across the body of the BAT along the beam's longitudinal axis. Two TV cameras are used, with one of them used to provide the operator with a "head-up" display, and the other to track the movement of the end effector. A communications link carries commands, telemetry, and video between BAT and the control station. A six-degrees-of-freedom arm with dexterity equivalent to the human arm is under development. This arm features a standardized wrist interface so that end effectors may be interchanged remotely under operator control.

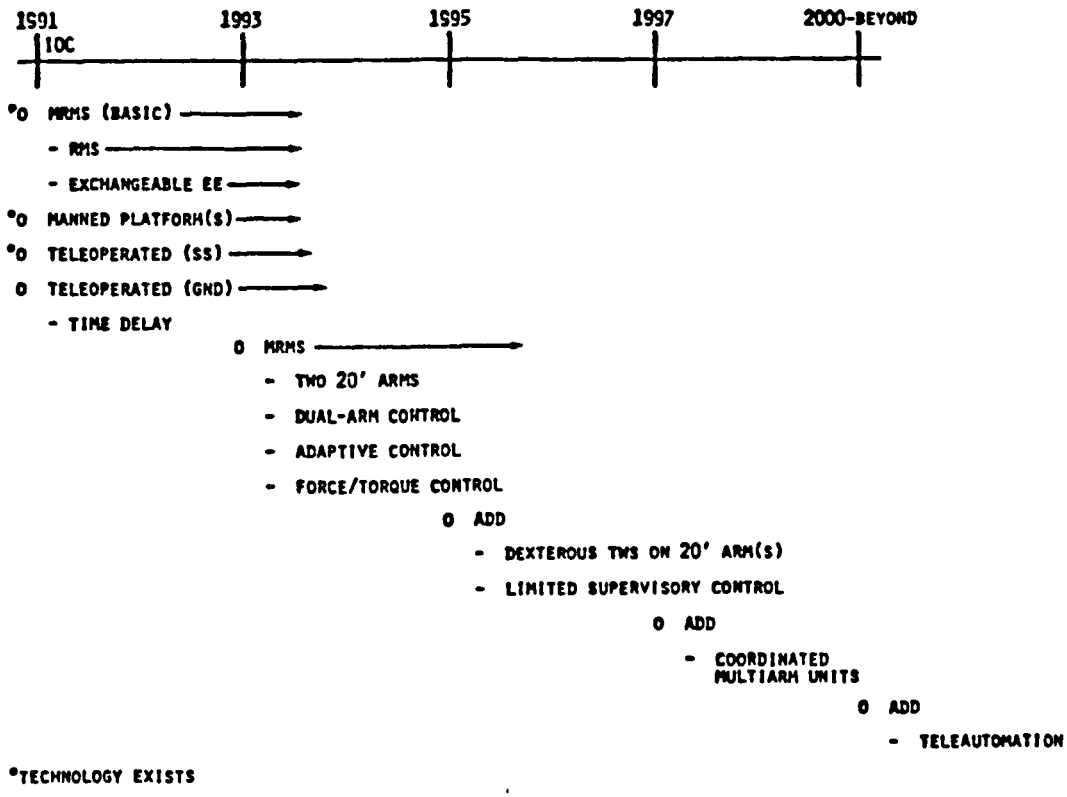


Figure 19: MRMS System Evolution (Martin)

(1) Manual control All manipulator actions are based on controller inputs.

(2) Aids to manual control. Additional sensing of worksite activity is achieved through force and tactile sensors.

(3) Supervisory mode. For single segments of a given task, the operator will have the capability to initiate a "supervisory" mode in which the computer has responsibility for executing the given task.

(4) Task specification mode. The operator specifies a class of tasks to be performed. The computer plans the task, including the sequence of activities, the selection of tools, and the handling of exceptions. The operator is notified only when workaround techniques fail.

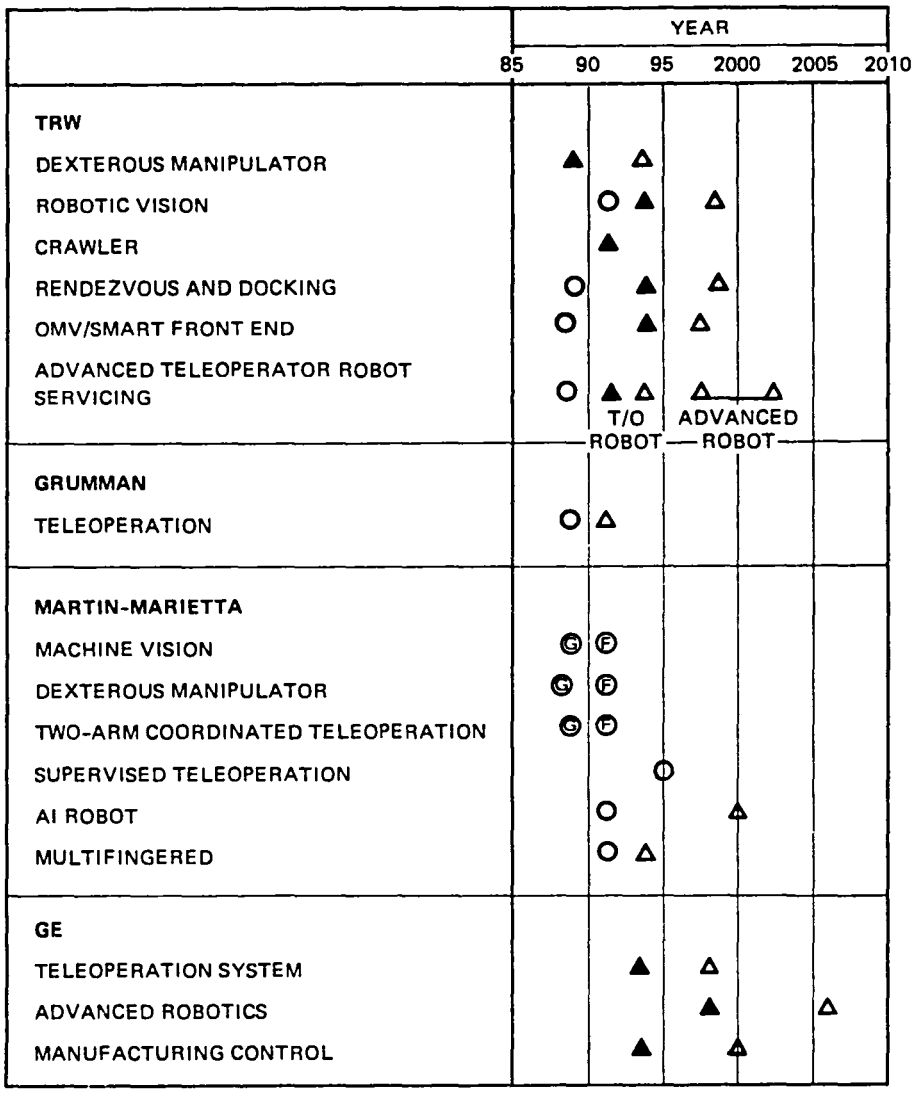


Figure 20: Coordinating the Contractors' Estimates

We can expect this type of evolution to be true for all teleoperation devices.

The consolidated chronological diagram of the required teleoperation capabilities is shown in Figure 21. In essence, it shows the following:

- 1990. Telepresence, with stereo vision, force reflection, a good arm with dexterous simple gripper.
- 1993. Same as above but with multifingered gripper.
- 1993-1995. Telepresence, with supervised autonomous subtasks, i.e., high-level commands by the crew member are converted automatically to the necessary low-level commands and executed with the aid of sensor information.
- 2000-2005. Robots, with self-contained vision, planning, and control, able to carry out manipulative operations autonomously.

2. *Sensors*

The need for sensors and perception arises as one attempts to accomplish more dexterous manipulation, and as one removes some of the detailed manipulations from the operator. Initially the sensed, interpreted data can be used to aid the operator by displaying the forces on a graphics display or by highlighting portions of the image that are pertinent to the operation. In later developments, this information can be used directly by the control system of the teleoperated device.

Sensors and perception are crucial to the "supervisory mode" and the "task specification mode" in which the computer controls a portion of the manipulation task. To accomplish this, the vision system must be able to send correction signals to the teleoperation control that indicate where objects are located, permits their automatic alignment, aids in the application of tools, etc.

Although some type of force feedback will obviously be required, it is not yet clear what tactile capabilities are needed to implement the various concept designs. For example, will it be important to sense the degree of slippage in a manipulator that is grasping an object? Will pressure sensing be required for fine manipulation?

Sensing and perception are discussed in more detail in Chapter 5.

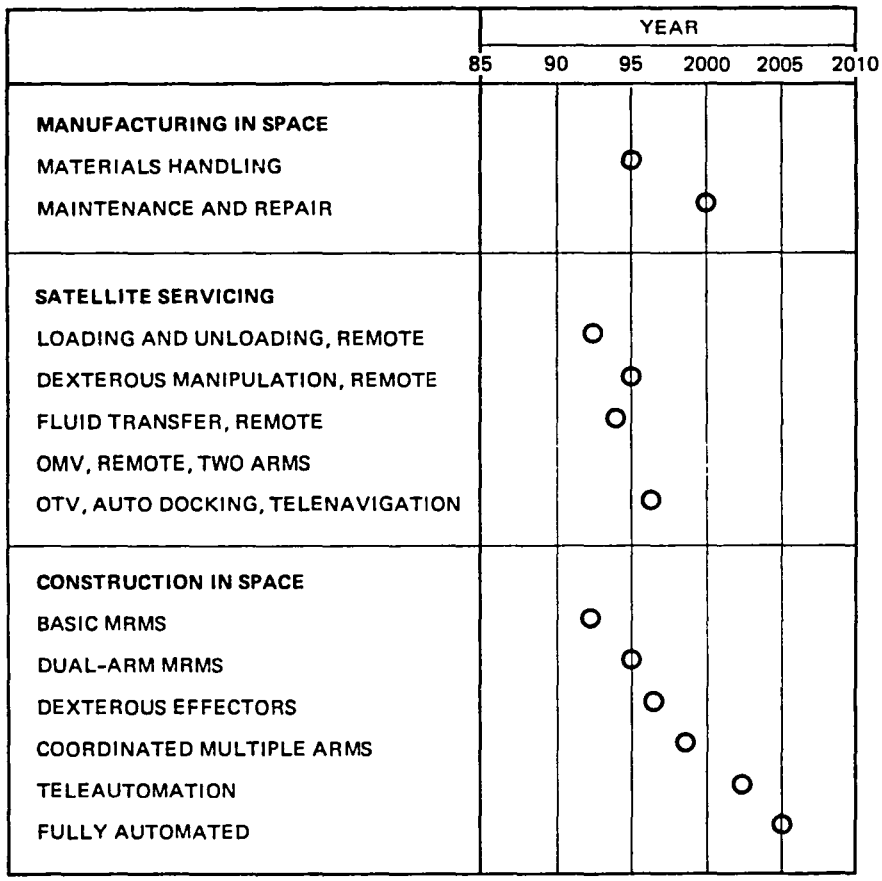


Figure 21: Implementation Schedule of the Teleoperation Capabilities

3. *Expert Systems*

Expert systems play a role in diagnosis and maintenance, in planning robotic movements, and in the control of manufacturing operations. Specifically, for the latter application GE identified the following quality assurance, process control, and maintenance roles for an expert process controller (EPiC):

- Quality Assurance: Assimilates process-monitoring inputs; identifies wafers for inspection; interprets process deficiencies; identifies the effects of corrective actions based on causal models.

- Process Control: Reviews effects of suggested process adjustments; determines best course of action; implements required process adjustments, tracks results; reconfigures automation timetable to accommodate process adjustments and reforecasts, schedule of

raw-materials usage and waste-materials disposal.

- Expert Maintenance Controller: Checks process equipment in response to equipment performance; flags abnormal transient operation prior to hard failure of process element; isolates equipment faults, troubleshoots, interprets best course of action in accordance with information from knowledge base.

Since many of these items have never been developed for earth-based manufacturing, considerable effort must be made to attain the foregoing EPiC capabilities.

Some of the general capabilities required in expert systems are indicated briefly below; expert systems themselves are discussed in detail in Chapter V.

- 1991-1992. Monitoring-and-diagnosis systems for selected space station subsystems capable of determining when trouble occurs and then identifying the problem.

- 1993-1995. More complex monitoring-and-diagnosis systems that can deal with interactions among systems and more subtle problems.

- 2000. System for space manufacturing capable of quality and process control, as well as maintenance of the manufacturing equipment.

The detailed timetable for implementing expert system capabilities is shown in Figure 22. We show the expert system for space manufacturing (EPiC) as being available during 1993-1995, and various systems designed to aid in maintenance as being available from IOC to 1997.

4. *Planning*

Some specific applications of automatic planning to the space station are listed below:

Automated Rendezvous. A system that can plan and replan the maneuvers of two or more vehicles that must rendezvous.

Astronaut Task Scheduling. A system that can plan and replan astronaut tasks.

Process Planning. A system that can plan the processing operations of a manufacturing unit.

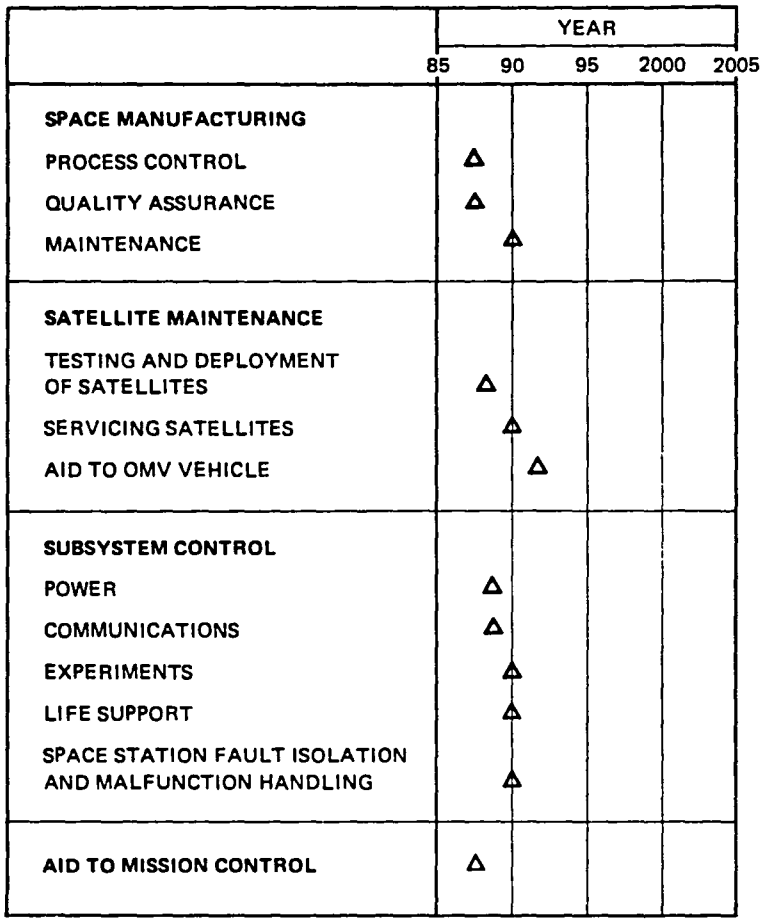


Figure 22: Sequencing of Expert System Capabilities

Adaptive Teleoperation. A system that can plan the submovements of a teleoperator arm and end effectors, based on high-level requirements specified by a human operator.

Autonomous Robots. A system that can plan the movements of an autonomous robot, taking into account the actions of other agents—both human and robotic.

The approximate timetable for automatic planning capabilities is as follows: (The details are given in the chapter on automatic planning.)

- 1993-1995. Planning subtasks to support telepresence capable of decomposing a high-level command into a lower-level set of actions.

- 2000. Planning of actions for maintenance and reconstruction of manufacturing equipment.

- 2000-2005. Planning for autonomous robots for navigation from one location to another and for cooperative work with other robots.

C. Application of Automation Technologies to Subsystems

Typical applications of automation technologies to space station subsystems are shown in the listing below. The important role played by teleoperation/robotics, sensors, and expert/planning systems can be seen clearly.

ELECTRICAL POWER

- **Teleoperation/Robotics**. Replacement of solar cells

- **Sensors**. Inspection of solar cell surfaces

- **Expert/Planning Systems**. Replanning of load distribution when failure occcurs

- **Planning**. Scheduling of solar cell maintenance and replacement

GUIDANCE, NAVIGATION, AND CONTROL.

- **Expert systems**. Onboard mission control, automation of traffic control, and trend analysis and identification, with indication as to corrective action needed

COMMUNICATIONS AND TRACKING

- **Teleoperation/Robotics**. Replacement of antennas and associated electronic packages

- **Expert systems/planning**. High-speed, dynamic scheduling and rescheduling of tasks when communication problems arise, antenna-pointing strategies, and automation of tracking, berthing, and docking.

DATA MANAGEMENT

- **Expert systems**. AI techniques applied to automatic programming, strategies for reconfiguration and fallback, and an astronaut "associate," (similar to the DARPA Pilot's Associate

concept)

- **Computer architecture.** Research leading to system that is distributed, fault-tolerant, expandable, and secure

PROPULSION

- **Teleoperation/Robotics.** Automated fuel transfer

ENVIRONMENTAL CONTROL AND LIFE SUPPORT

- **Expert systems.** Identification of trends and suggestions regarding actions to be taken, and gas analysis to determine malfunction; and suggestions as to its cause and how to rectify the trouble.

THERMAL CONTROL

- **Teleoperation/robotics.** Repair and maintenance of surface coatings

STRUCTURES AND MECHANISMS

- **Teleoperation, sensors, expert systems.** Automated berthing, and construction of space structures

HABITABILITY

- **Expert systems.** In-flight health-care system

- **Natural language/speech.** Voice input/output

PAYLOADS

- **Expert systems.** Trend analysis and control of experiments

PLATFORMS

- **Expert systems.** Control of manufacturing processes; diagnosis of manufacturing problems

- **Teleoperation/robotics.** Materials handling, maintenance, and repair

VERIFICATION

- **Expert systems** System checkout, verification, and recovery

D. Availability of Other Research

Automation research by others may yield some of the results needed by NASA—but not early enough for some critical automation topics. In particular, results in teleoperation/robotics may not apply to a zero-gravity space environment. The main non-NASA sources for automation research are the following.

1. *Department of Defense*

DARPA's SCP [3] will produce advanced computer architectures and system-level software, rather than manipulators and dexterous grippers. DARPA is not interested in teleoperation or in the special problems posed by a zero-gravity environment. Its main interest is in aids to human decision-making and in automatic navigation for land-mobile robots. DARPA's only navigation problem, however, is to find a route over an essentially *two-dimensional surface* populated with three-dimensional obstacles that block the view of sensors. NASA, on the other hand, requires *three-dimensional* route-planning algorithms to get around three-dimensional obstacles. The case of navigating within the station itself or inside a large orbital structure is particularly difficult, because the robot will usually be completely surrounded by obstacles.

The DARPA Autonomous Land Vehicle (ALV) project includes a considerable amount of vision research dealing with terrestrial scenes, especially roads. The results in this area may not be very applicable to NASA's space environment problems. First, DARPA needs *passive* sensing methods for a low signature. NASA, on the other hand, can (and should) use *active* sensing—structured illumination, in particular— to extract three-dimensional shape information, make precise measurements, reduce contrast in bright sunlight, and disambiguate scenes to simplify interpretation. Furthermore, DARPA's robots will always view the terrain from a point on or near its surface, and will have the directing influence of gravity to exert a strong disambiguating influence on scene interpretation. Some of DARPA's research results relating to this kind of route planning could be useful to a NASA robot navigating along the outer surface of the space station.

Unlike the ALV situation, however, NASA's robots and the structures they navigate in and around can float freely. Thus, they will have to be able to recognize their "terrain" from *any* point of view, and will not have the advantage of gravity to help them guess how things "ought" to appear. Furthermore, NASA's robots will operate in a highly structured and controllable world whose every dimension has been entered into a CAD data base. They will therefore be able to routinely apply vision and navigation strategies that are not applicable to DARPA's robots. To ascertain their position, for example, NASA's robots will probably be able to rely on known "survey markers" (e g., lights, signs, bar codes)

placed at numerous locations on space structures.

DARPA's Engineering Applications Office is in the early stages of planning a new initiative concerned with modular, repairable robots, expert systems for automated repair, and design for easy repairability. The research results of this effort should be very important to NASA, particularly for satellite servicing. There may eventually be additional spinoff benefits for other space applications that are not immediately foreseeable.

DARPA's Intelligent Task Automation (ITA) project [4], [5], a joint industry-university research effort in cooperation with the U.S. Air Force, has as its goal the reduction of the *task-specific* portion of the software needed to make a robot assemble a particular product. Since this is the part that changes from product to product, its smaller size should result in decreased programming cost per product, hence lower manufacturing costs. It will also tend to diminish the size of an economic batch size—an important cost factor in optimizing military replacement parts. Pertinent research includes 3-D vision (dense and sparse range sensing), driven by CAD models of object shape, and multisensory integration (vision and force/torque). It is directly relevant to the adaptive-robotics level of autonomy for space automation systems, as described in this report.

Other military research in automation includes the following:

- *U.S. Army.* Land-mobile systems for handling large items of materiel, such as shells and fuel drums. Little interest in dexterous manipulation for repair of equipment.

- *U.S. Navy.* Autonomous underwater mobile systems, probably not applicable.

- *U.S. Air Force.* Manufacturing technology to reduce costs, including Integrated Computer-Aided Manufacturing (ICAM) Project, and Intelligent Task Automation (ITA) Project with DARPA. The most likely contribution to space station automation will be CAD-based fusion of visual, inertial, and tactile sensory information.

2. *Other Sources*

Other sources of research:

- *National Bureau of Standards (NBS).* The NBS has a project comparable in scope to the U.S. Air Force's ICAM program.

- *National Science Foundation (NSF).* Funds basic research on

many important core topics in automation, but cannot support projects at the level of effort needed by NASA.

- *University.* Universities have provided 90% of the automation technology that NASA could obtain today. However, their primary mission is to educate students -- a constantly fluctuating personnel component that makes it very difficult for academic institutions to carry out highly organized, tightly scheduled multiyear programs. Though universities have produced many important software systems, some of which represent major technological breakthroughs, these usually are the inspired work of one, or at most a few, extremely talented individuals guided by frequent, informal interaction with their peers.

- *Commercial.* Commercial firms focus on immediate, usually low-technology solutions to their immediate manufacturing problems. They tend to take technology developed previously in universities. Recently most of the larger corporations have set up in-house research groups on advanced robotics and artificial intelligence. These will begin to bear fruit in the next two to five years. A few may even rival academic centers in performance. However, because industry is rapidly recognizing that these technologies will be critical for their very survival as corporations against foreign competition, they will jealously protect their proprietary interests. Since transfer of technology to NASA is viewed as tantamount to placing it in the public domain, NASA may have to adopt new policies to obtain private industry's full cooperation.

The balance of this report deals with the AI-based technologies required to achieve the capabilities we have been discussing, postulates demonstrations that illustrate these capabilities, and indicates the research and development needed before these demonstrations can actually be realized. In doing this, we have expanded beyond the limited context of the capabilities alone, as they are identified in the contractor's concept designs.

REFERENCES

1. E.B. Shain, "Design and Control of A Beam Assembly Teleoperator (BAT)," (Report SSL#24-83), Cambridge, Mass., Space Systems Laboratory, MIT, May 1983. Sponsored by NASA MSFC Contract NAGW-21.

2. "Analysis of Remote Operating Systems for Space-Based Servicing Operations," (Final Briefing SA-ROS-RP-10), Bethpage, New York, Grumman Aerospace Corporation, November 1984. Performed for NASA Johnson Space Center.

3. "Strategic Computing, New-Genertion Computing Technology: A Strategic Plan for its development and Application to Critical Problems in Defense," (Tech. Rep.), Washington, D.C., Defense Advanced Research Projects Agency, October 1983.

4. "Intelligent Task Automation," (Quarterly Reports), Denver, CO, Martin Marietta, 1984.

5. "Intelligent Task Automation," (Quarterly reports), Roseville MN 55113, Honeywell Inc., 1984.

IV. DESIGN FOR AUTOMATION

IV DESIGN FOR AUTOMATION

Industry is slowly learning how to employ techniques such as chamfering [1], mechanical compliance, and lighting engineering so they can use simpler, less accurate robotic equipment. Unfortunateley, there is at present no complete, formal, rigorous theory of design for automation—only ad hoc design principles and rules of thumb. For example, although we know that "parts should be designed to be easy to identify automatically," we have no way to decide if it would be best to give a particular part a distinctive shape, stencil its name on the top, attach a bar code label, paint it a special color, or mold a binary notch pattern into one edge. Most present-day CAD/CAM data bases represent design information in forms unsuitable for answering this and other important questions, such as how to interpret sensor data or how to disassemble and reassemble a piece of equipment.

What little theory of design for automation has been developed so far concerns evaluating the suitability of a part's shape for automatic feeding, orienting and assembly, and the use of compliance to prevent jamming during insertion. The zero gravity and vacuum of the space environment will require additional research to decide how to design for automation under these unique conditions.

A. Space Station Applications

In designing for automation, one identifies and provides specific physical accommodations that must be included as part of the initial operational capability (IOC) of the space station*. Their purpose is to simplify the operation, diagnosis, and repair of space station equipment, as well as to make it easier to automate these tasks. Another major consideration is to provide for increased *future* automation of the space station and, with this in mind, to avoid limiting NASA's options by unwise, premature design choices; flexibility and the potential for growth in power and scope are crucial factors. The key accommodations are for the computing requirements of expert and planning systems and for making equipment repairable by teleoperation/robotics.

---

*Sometimes called "hooks" (software design features) and "scars" (hardware design features)

1. *Accommodations for Expert and Planning Systems*

We envision four types of computer systems as necessary to support the variety of expert and planning systems needed for the space station:

(1) Astronaut work stations with graphics that have both conventional- and symbolic-programming-language capability.

(2) Space-qualified symbolic processors connected via a network interface to system data bases.

(3) Space-qualified symbolic processors connected via high-speed interfaces to sensors, switches, and other essential control points of equipment and subsystems.

(4) Portable symbolic-processing systems that require no interface with any subsystem or equipment.

Accommodation must be made for the first three systems by providing access to the space station data base and by requiring that subsystems make sensor and control points available.

2. *Accommodations for Automation*

Setting up a design data base for all equipment on the space station (including automation equipment) is an important accommodation suggested by most of the contractors. Thus, in developing space station systems and equipment, it is critically necessary that information about these systems be incorporated in a system data base. Such information should include CAD/CAM specifications, structural and functional descriptions, and as much design information as possible. Operating and maintenance procedures should also be in the data base, including annotations describing the purpose of every routine and of each step within each routine.

Other accommodational concepts fall into the categories of (1) design of the automation equipment itself and (2) design of other space station equipment:

<u>Design of Automation Equipment</u>

- Modular, self-repairable manipulators, capable of being assembled manually, automatically, or by teleoperation to create different sizes and configurations of arms to serve different purposes. A modular, self-repairable manipulator has already been developed by Oak Ridge [3] for the nuclear industry.

- Hhigher-sensitivity force-reflection—e.g., by using a new direct-

drive joint design with extremely low friction telepresence operations [4]. It uses a new kind of electric motor with low enough speed and high enough torque to be connected directly to the driven segment of the arm. Gears, pulleys, leadscrews, etc. are eliminated. Such a joint has only bearing friction, which is at least two orders of magnitude less than the toal friction in a conventional joint. This would greatly improve dexterity for handling small parts or delicate equipment on the space station.

- Redundant components in the rotating joints of manipulators for greater reliability (rotating joints are inherently unreliable in space).

- Digital communication network with excess capacity connecting space station equipment with computers that also have excess capacity.

- A family of general-purpose equipment connectors (GPECs) in different sizes that would provide sturdy mechanical attachment and support, power, and access to the space station's local-area network. These could not only be distributed over the space station's structure, but could also be built into any conventional or automation equipment.

Design of Other Space Station Equipment

- GPECs to connect to the space station and other equipment.

- Simple hard points or holes that teleoperator/robotic equipment could grasp in order to stabilize itself with respect to a satellite or a work area on the space station. They could also be used as "footholds" for walking along a space structure.

- Design for easy location, identification, handling, and servicing by imprecise automation equipment operating in zero gravity.

B. Demonstrations

The Space Operations Mechanism Testbed at the NASA Marshall Space Flight Center (MSFC) will have a pivotal role in matching the appropriate technology with evolving space station requirements, and evaluating alternative mechanisms in an integrated manner. The following demonstrations indicate a capability of designing equipment so that it can be handled and repaired by automated devices:

- A data base organization that can be expanded to handle information that may be necessary for future space station operation, including information about the organization, contents, and access methods of the data base itself.

- A prototype data base that can describe the structure, method of operation, and maintenance procedures for (1) simple mechanisms constructed mainly from rigid parts, (2) electronic equipment, (3) complex electromechanical-hydraulic equipment, and (4) space station subsystems.

- A methodology for verifying that a piece of equipment can be serviced and repaired successfully by astronauts in EVA, by teleoperation, and by robots.

- A family of mutually compatible, space-qualified, general-purpose, modular sensors, effectors, controls, and connective elements for the rapid construction of specialized devices, including teleoperators and robots, by other teleoperators and robots, or by humans.

REFERENCES

1. D.E. Whitney, R.E. Gustavson, and M.P. Hennessey, "Designing Chamfers," *The International Journal of Robotics Research*, Winter 1983, 2(4), pp. 3-18.

2. V.R Hamel, M.J. Feldman, and H.L. Martin, "Advanced Teleoperation in Nuclear Applications," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 302-305, Las Vegas, Nevada (12-15 August 1984).

3. H. Asada and H. Yamamoto, "Torque Feedback Control of MIT Direct-Drive Robot," *Proceedings of the 14 th International Symposium on Industrial Robots*, pp. 663-670, Göthenburg, Sweden (2-4 October 1984).

V. TELEOPERATION AND ROBOTICS

V TELEOPERATION AND ROBOTICS

This chapter discusses the teleoperation and robotics requirements for satellite servicing, manufacturing in space, and orbital construction.

A. Introduction

The TRW study of satellite servicing concludes:

> *"Telepresence is the principal automation discipline required for servicing, with human operator involvement to handle task diversity and unforeseen situations."* [1]

The TRW report indicates the need for (1) both gross and dexterous manipulation, (2) the ability to deal with flexible objects, (3) the ability to execute "learned" sequences, and (4) multiarm operation. The GE manufacturing study identified refurbishment of the furnace by robotic disassembly as the biggest challenge to be overcome [2]. In addition, process room robots would have to exchange modules and transfer cassettes. Martin Marietta described a "multipurpose logistics device" outfitted with a space crane and EVA positioning arms, a tool to transport modules and/or payloads from the shuttle cargo bay and position them for attachment to the space station truss structure [3].

Below we treat the important subject of teleoperation and robotics. To clarify the discussion, we have used an antenna deployment task as the basis for scenarios that describe how the task would be accomplished by means of (1) telepresence, (2) a preprogrammed adaptive robot, and (3) a self-programming "intelligent" robot, in Appendix B. These scenarios provide a basis for determining the functional requirements for each level of automation.

Teleoperation and robotics reflect a broad spectrum of important automation concepts for the space station—from very low to very high levels of autonomy.

- *Teleoperation (TO)* is remote manual control of equipment that is capable of sensing, manipulation, and/or mobility. The equipment may be only a few feet from the operator or it may be as far away as another planet.

- *Telepresence (TP)* is an advanced form of teleoperation in which feedback of visual, tactile, auditory, or other sensory information

from a remote work site gives the illusion of being there so that the operator can exercise better, more precise control.*

- *Supervisory control (SC)* or *augmented teleoperation (AT)* is a mixture of manual and automatic control modes. As the amount of automatic control increases, it begins to approximate adaptive robotics.

- *Adaptive robotics (AR)* is full automatic control of the equipment by a computer in accordance with a program that makes it react in predetermined ways to data from sensors that report external conditions [4].

- *Intelligent robotics (IR)* is adaptive robotics in which AI-based reasoning and planning programs develop the detailed control steps, either to carry out high-level instructions from people or to respond creatively to unforeseen conditions and events during a mission.

Space station automation for servicing, manufacturing, and construction tasks should evolve in the direction of increasing autonomy [5]. It should start at the telepresence level, rather than teleoperation, because modern microprocessor technology can provide much additional functional utility for a very small weight penalty. The evolution sequence should be from telepresence through supervisory control, and adaptive robotics to the long-term goal of intelligent robots.

An intelligent robot should be designed so that it can be operated as a preprogrammed adaptive robot or telepresence system whenever its AI software is unable to perform a particular task successfully. Similarly, an adaptive robot should be remotely controllable in order to deal with situations that are too difficult for preprogramming. Providing these *fallback modes of operation* will make the overall robotic system more robust and reliable.

The intelligent robot stage of evolution will probably occur in two steps: Initially, AI software will generate robot control programs for "unintelligent" adaptive robots to execute. This would be an "off-line" process, and at first would probably involve close interaction with a person through a sophisticated, multimodal programmer's workstation environment. Later, a more closely-

---

*In this report, we assume that NASA will want to take advantage of the benefits of advanced telepresence methods wherever practical. Therefore, we use the latter term in place of teleoperation or teleoperation/telepresence to encourage the reader to think about space station automation in the manner thereby suggested.

coupled arrangement will develop in which AI software, perhaps on-board, operates the robot(s) directly, interacting only minimally with people to obtain task assignments. For simplicity, we do not distinguish between these two steps in this report.

CAD-based simulation of teleoperator and robot activity will play an important role in most stages of this evolution. Such simulation would use computer models of the objects, the environment, and arm dynamics to predict the motion of the arm from the operator's control signals [6]. NASA has already taken steps in this direction, developing at least one such simulator [7] at Langley. At Goddard, they have extended the National Bureau of Standard's Integrated Graphics Exchange Standard (IGES) to support three-dimensional solid modeling by constructive solid geometry [8], developed a PROLOG-based automatic planner for robot manipulator motions that is driven by such models [9], and captured geometric information for the experiment packages on the Gamma-Ray Observatory in a CAD data base [10].

CAD-based simulations will first be used to make remote control easier in the presence of moderate (e g., one second) communication delays between master and slave. For example, as an operator moves the master control of a manipulator, a computer will display a synthesized picture of the slave arm moving *without delay* in response to the control signals. This *predictive display* will allow the operator to move the arms more smoothly and rapidly, without waiting for television images to arrive from the slave equipment. When an operator is remotely steering a mobile robot, the predictive display shows the path that the robot will travel during the round-trip communication delay time [11].* The same simulation technology will also be useful later in advanced workstations needed for ground and space station crew programming of adaptive robots and other automation systems. The crew will not require extensive training in formal computer programming languages, but will operate a simulation of the robot or other automation system to illustrate the procedure to be carried out. Finally, the ability to simulate the activity of a robot performing a task will be absolutely vital for automatic planning of robot activity by artificial intelligence techniques. The simulation will allow the planning program to evaluate different action sequences to find a feasible or even optimal procedure for accomplishing a task.

Effective telepresence systems must allow the operator to see and feel objects at the remote work site. An important technique will be *master-slave*

---

*The question of what to do when the remote system fails to perform as predicted is an important and challenging research problem. For example, it may be necessary to preprogram the remote system with "protective reflexes" that can automatically place it in a safe condition until it receives new instructions from its operator.

control, in which the operator moves the end of a "master" control arm and the remote, or "slave," arm follows it. A *replica master* is usually a scale model of the slave, with the same shape and kinematic arrangement of joints. However, a replica master can have a completely different shape and joint arrangement if a computer is used to convert its joint motions into equivalent motions of the slave's joints [12].

Automatic activities can be programmed *procedurally*—e.g., by recording teleoperated activities for later replay, or by more complex training processes [13]. This method will be particularly useful in *supervisory control* of simple manipulation tasks [14, 15]. More complex tasks will require a *textual* programming method, such as construction of a robot control program in a formal programming language [16, 17]. The system then becomes a preprogrammed *adaptive robot* that can perceive and react to external events and conditions.

An *artificially intelligent robot* might look very similar to the telepresence system or adaptive robot, and indeed might be developed from them. The main physical difference would probably be a much more powerful computer system onboard the robot (if it is not remotely controlled by the space station's central computers). From a functional standpoint, an intelligent robot would be far more robust and reliable than a preprogrammed adaptive robot because it could *reprogram itself* to respond to changing conditions We must caution the reader, however, that such a system is far beyond the present state of the art.

B. State of the Art

The technology of telepresence and robotics is already sufficiently developed to perform many important perceptual, manipulative, and mobility functions on the space station. In the future, artificial intelligence will be needed for planning manipulator actions, interpreting sensor readings, and navigating around and within complex structures.

1. *Perception*

Perception technology today is adequate only for highly controlled and predictable situations, such as manufacturing and simple satellite servicing. In orbital construction tasks the bright sunlight will tend to degrade important details, and constantly changing shadows will make it difficult to ascertain the true shape of an object. Free-flying objects may appear in any orientation, creating a difficult recognition problem. Variations in the shape of nonrigid components and equipment due to major structural damage will also make shapes more difficult to recognize. Although three-dimensional imagery eliminates much of the complex processing that conventional two-dimensional imagery requires, its interpretation is still difficult. Tactile identification of an object based on how it

"feels" to a wrist force/torque sensor, a tactile array, or the fingers of a dexterous hand is currently in the research stage [18, 19, 20]. Finally, not enough work has been done on using CAD models of objects to facilitate visual and tactile perception [21].

2. *Manipulation*

Manipulation technology is well developed for many manufacturing and satellite-servicing tasks on the space station that can be performed by relatively *short* manipulators (e.g., less than two meters long) moving at a relatively slow speed (e.g., less than one meter/second) with low accelerations (e.g., less than one gram). Manipulators of this size can be made with low enough mass for space station use, yet stiff enough so they will not flex excessively as they move. Adequate control methods have been developed for single arms of this type mounted on a rigid foundation. But if such an arm were mounted on a cherry picker or free flyer, there could be new potential for instability and destructive oscillation.

Construction tasks in space require a rather *long* manipulator [22]. But, since it is large, it is important to minimize its mass to reduce boost and reboost costs [23]. That causes a control problem, because the individual links that connect its joints are likely to be quite flexible, even if constructed of advanced materials such as fiber composites or titanium alloys. Any flexibility *between the joints* makes rapid, accurate motion much more difficult to achieve. In fact, moving a long, limber arm rapidly and accurately—especially when it is carrying a relatively massive load—is quite beyond the state of the art of control theory today, though some progress has been made on the problem [24]. Fortunately, in construction tasks, slow, careful motion may be quite adequate; indeed, it may be much safer.

Industrial market pressures are driving the development of higher-performance manipulator hardware and unusual kinematic configurations are beginning to appear. "Tentacles" have been developed for using grippers and tools in confined spaces [25]. Parallel-jointed arm designs offer both high strength and accuracy. In the near future we may expect micromanipulators for delicate repairs and perhaps dendritic (multiple-branched) manipulators for complex construction and assembly tasks.

3. *Mobility*

Mobility technology today is quite adequate for manufacturing and satellite servicing tasks, but needs improvement for orbital construction. Simple rail or cable transport systems would satisfy manufacturing needs, while the reaction propulsion methods for free-flying satellite servicers reached a high degree of development years ago. More innovative methods, such as ducted-fan propulsion

and legged locomotion await additional research. Several automatic "walking" robots equipped with legs have already been demonstrated [27], and, reportedly, NASA has recently purchased a six-legged teleoperated one.* However, these devices cannot yet see the shape of the surface they walk on, steer around obstacles automatically, or select their own footholds. Although mechanical legs can be controlled by applying manipulator control methods in a straightforward manner, certain problems related to coordinating multiple legs and preventing dynamic instability (primarily in a gravitational field) are still topics for research [29, 30, 31].

Mobility will be an important capability for many automation systems—e.g., to carry materials during construction, capture satellites, service co-orbiting platforms, and inspect large, delicate tension structures such as mesh antennas. The most important modes of locomotion for use on the space station include rail transport, crawling [32, 33], and free flight. Rail transport is simple and need not expend consumables, but the rails add weight in proportion to the size of the space station, can become blocked, and allow only limited motion. Crawling need not expend consumables, carries a constant weight penalty for any size station, and allows free motion over the surface and within structures, but it requires relatively complicated equipment. Free flight allows the most freedom of motion, but requires moderately complicated equipment, consumes fuel and reaction mass and may create plume impingement problems, and carries a constant weight penalty. For free flight within pressurized station modules, aerodynamic propulsion methods such as ducted fans are possible.

C. Pacesetter Technologies

This section discusses the pacesetter technologies for space station teleoperation and robotics. In many cases, the levels of performance currently available are seriously deficient for NASA's purposes and will have to be improved considerably to be of any eventual use in the space station. The discussion is organized according to the hierarchy shown in Figure 23. This figure groups pacesetter technologies into two low-level categories of equipment (input and output), three intermediate-level ones (interpretation, generation, and adaptive control) having to do with operation of the equipment, and a high-level category (reasoning) in which artificial-intelligence methods are employed to decide what to do with the equipment. Since man-machine interface technology cuts across the top two levels in the figure, we discuss it separately in Section V-C-5 below. Although some of the technologies treated there are also treated in other chapters, we felt that it was important to indicate how they must

*An ODEX-I "functionoid," manufactured by Odetics, Incorporated, Anaheim, California [28]

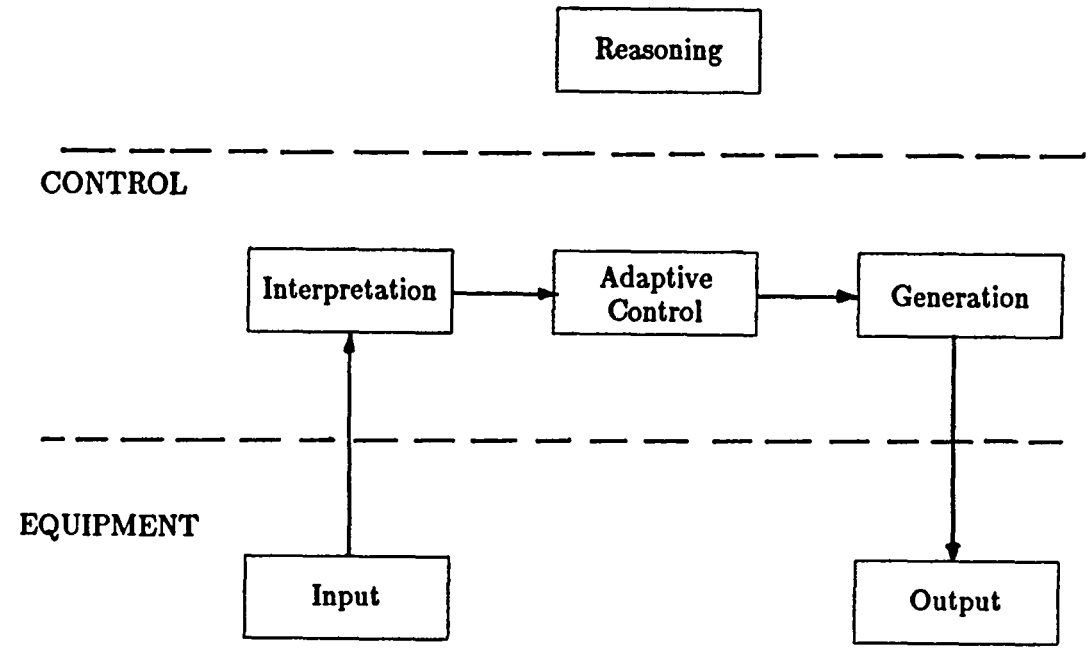


Figure 23: Pacesetter Technologies for Teleoperation and Robotics

be integrated to function smoothly together in teleoperators and robots.

It should be emphasized that a comprehensive data base describing the construction and operation of the space station will be extremely useful—and in some cases crucial—for many of the teleoperation and robotic technologies. The more detailed this data base, the more useful it will be.* These data must be acquired as the initial operational capability equipment is being designed and constructed, and not afterwards.

The following sections discuss each of the categories in Figure 23, page 69. Appendix A lists approximately 90 specific examples of equipment and techniques related to these six categories.

*Down to the level of individual components, if possible.

1. *Equipment*

Input transducers, or *sensors*, such as cameras and tactile arrays can easily produce information faster than current algorithms running in current computers can interpret it. The opposite situation holds for output transducers, or *effectors*: they usually cannot respond to commands as fast as the software can produce them. Much of the theory related to dexterous manipulation and to locomotion on legs has never been put into practice, because poor performance of the motors and mechanisms available to drive limblike appendages prevents the use of sophisticated control algorithms.

The requirement of space qualification is a general problem for most equipment needed for teleoperation and robotics on the space station. Qualification of systems to be used only inside the space station, in a specially created, "shirtsleeve" environment is considerably easier to achieve.

The pacing technologies for *input* equipment performance are sensors with improved physical characteristics, new sensors to measure important physical quantities for navigation and manipulator control, and improved equipment for people to communicate with automation systems.

The pacing technologies for *output* equipment performance are actuators with improved physical characteristics, combinations of actuators with mechanisms to produce new or improved limbs and end effectors for robots, and improved equipment for automation systems to communicate with people and other automation systems.

2. *Control*

In our paradigm (Figure 23, 69), *interpretation* of data from the input equipment drives *adaptive-control* algorithms to *generate* the control signals that operate the output equipment.

Some of the pacing technologies for *interpretation* of signals from sensors are the use of CAD data and other kinds of knowledge bases to aid interpretation, "smart sensors," automatic analysis of mechanism and free-body dynamics, tactile signal interpretation, and multisensory integration.

Adaptive control is the link between the interpretation of sensor data and the generation of effector control signals, combining the current interpretation of the raw sensor data with a record of preceding events and the current "world model" to arrive at a situation assessment. Then, according to a preprogrammed algorithm, it selects an appropriate response to be generated. It makes use of such processes as geometric modeling, model-matching [34], transformation of coordinates, trajectory planning [35, 36], collision avoidance [37, 38], degree-of-

freedom analysis, motion coordination, compliance control [39], and symbolic-programming methods (i.e., use of a programming language).

Some of the pacesetter technologies for adaptive control are the use of CAD data and other knowledge bases in situation assessment, methods for specifying and representing the adaptive control algorithm, rapid numerical computation algorithms for planning motions, specialized task-related control algorithms, and generalized multisensory integration methods.

In telepresence and robotics for the space station, *generation* of control signals to operate output devices is mainly the use of feedback control methods to operate electrically-powered mechanisms. For adequate performance, some of the wide-bandwidth feedback signals may have to come directly from the input equipment, bypassing the adaptive control level of Figure 23 through a path not shown. Well-known, classical servo control methods will suffice to operate much space station automation equipment, but flexible or high-speed manipulators will require more advanced methods.

3. *Reasoning*

Reasoning, used more in robotics than in telepresence, includes the various categories of artificial intelligence: Image understanding, natural language, expert systems, and automatic planning. It makes use of processes such as logical deduction, probabilistic inference, search among alternatives, hypothesis formation, temporal reasoning, spatial reasoning, pattern matching, and learning. Some of the pacesetter technologies include geometric reasoning, commonsense reasoning, planning complicated procedures, expert systems for situation assessment, methods for dealing with failure to complete a procedure successfully, and machine learning.

4. *Man-Machine Interface*

In *man-machine interface* technologies we include teleoperator controls, the remote sensing methods of telepresence, and methods for specifying activities to be carried out automatically. Natural language, a category of artificial intelligence, will have its major impact in this area of space station automation.

a. Pacesetter Technologies

Some of the pacesetter technologies related to the man-machine interface are the use of CAD and other knowledge bases to aid the operator of a telepresence system or the programmer of a robot, telepresence master controls suitable for use in space, and higher-performance slave equipment to control and robotic equipment to program.

b. Milestones

Some important milestones for space station automation efforts in the man-machine interface area include the following:

- Hand coding of adaptive robot control programs by a person skilled in computer programming, robotics, and the particular space station procedure that the robot is to perform.

- Semiautomatic generation of the robot control program by another program from information it obtains by interacting with a person skilled in robotics and the particular procedure.

- Automatic generation of the robot control program by an artificial intelligence program from a description of the task supplied by a person skilled in the procedure, and the space station data base.

- Direct real-time operation of the robot machinery by artificial intelligence software.

c. Tactile Feedback

While force reflection has been a common feature of commercial teleoperator systems for many years, methods for presenting other kinds of tactile data to a human operator are still mostly in the research stage. Now that small, rugged tactile array sensors are becoming available for industrial robots, it will be easier for NASA to advance this interface technology. There is a large body of useful literature from the rehabilitation community on the topic of tactile presentation methods, and at least one commercial device for high-resolution tactile stimulation of human fingertips [40, 41].

It is possible to display the net force and torque acting on a robot hand graphically on a display screen. A recent experiment at JSC, for example, used bar-graph displays of the three force and three torque components [42]. Of course, the operator would require some training and practice to be able to interpret such displays. More intuitive graphical display formats are being considered at JPL. Graphic display of force and torque information would allow the operator's control to be a very small joystick, because it would not need to exert any force on the operator's hand [43, 44, 45]. A six-degree-of-freedom joystick can easily be made from a commercial six-degree-of-freedom wrist force/torque sensor.

Any of the following kinds of tactile feedback from the gripper could be provided, if necessary:

- Indication of contact or no contact with an object between the fingers.
- Indication of contact with each finger separately.
- Indication of slippage of the fingers with respect to the object they are holding.
- Proximity to objects within a few inches of the gripper.
- Grasping force [46].
- Grasping force at each finger (may be different—e.g., when the hand is not centered on the object as the fingers close).
- Vector force acting on each finger.
- Pressure distribution over the surface of each finger.

Each of the above could be presented graphically to the operator, but a kinesthetic presentation would probably be much easier to interpret and would probably result in better dexterity. For example, grasping force could be reflected through an actuator in a "plier" hand grip.

D. NASA Telepresence/Robotics Research

This section describes recent work by NASA and its contractors in teleoperation/robotics. We begin with a discussion of design concepts for satellite servicing systems, then review significant telepresence, supervisory control, and robotics experiments. Finally, we describe several of NASA's facilities for research in these areas.

1. *Satellite Servicing System Concepts*

In addition to the RMS [47], which is most suitable for simple handling, NASA and its contractors have developed several telepresence concepts for servicing the space station and satellites. The most important are the Remote Orbital Servicing System (ROSS), Figure 24, by Martin-Marietta [48, 49] and the Teleoperator Work Station (TWS), Figure 25, by Grumman [50, 51 pp. 76-78]. These two devices both have teleoperated manipulators and a pair of stereo television cameras in positions corresponding to the operator's eyes. However, despite the superficial similarity, the TWS, with its waist and neck joints, is more highly articulated than the ROSS. When the operator uses a head-mounted display, the "neck" would allow the TWS to move the cameras to follow the operator's direction of gaze. Since we use the ROSS in a scenario in Appendix B,

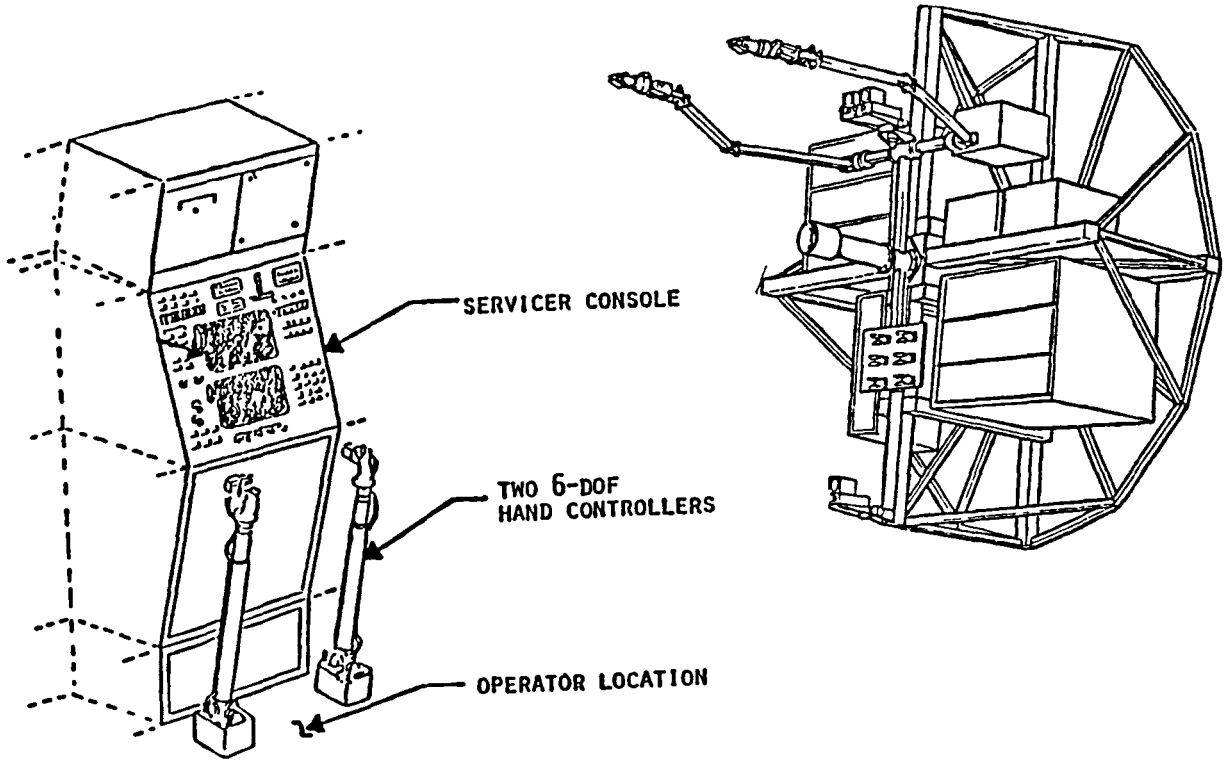
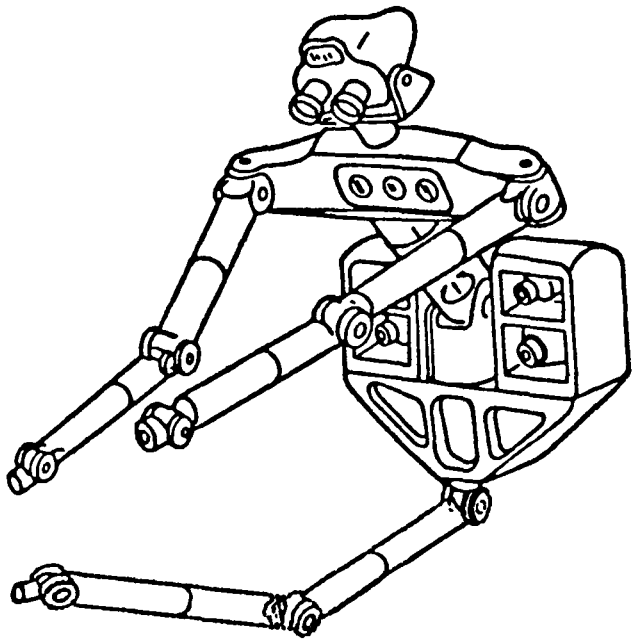


Figure 24: Remotely-controlled Orbital Servicing System (ROSS)

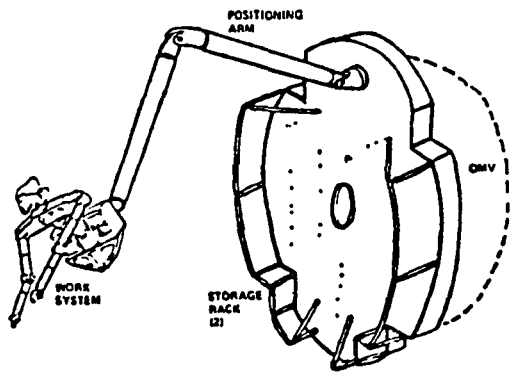
we will describe it more fully here.

The ROSS is a design concept for a general-purpose, multiarmed, camera-equipped telepresence system with bilateral force feedback (i.e., a force-reflecting teleoperator). A. J. Meintel's group at the NASA-Langley Research Center, as well as other groups in NASA, use it as a goal to focus their automation research [7]. Roger Schappell's group at Martin-Marietta developed the ROSS concept for NASA. It consists of an octagonal frame that carries two (or in some configurations, four) electrically-operated "slave" arms, a number of cameras, and a standard docking probe. The ROSS can be transported by another piece of general-purpose space equipment called the Orbital Maneuvering Vehicle (OMV), a teleoperated free-flying platform guided by remote control. The OMV provides the ROSS with power as well as radio transmission and reception services.

The ROSS is equipped with various sensors—cameras, joint torque sensors, and perhaps some kind of tactile sensors in the wrists or grippers. Two of the cameras are positioned for stereo vision, and others may be mounted on the frame



SYSTEM CONCEPT - FREE FLYER SERVICE



SYSTEM CONCEPT - SPACE STATION SERVICE

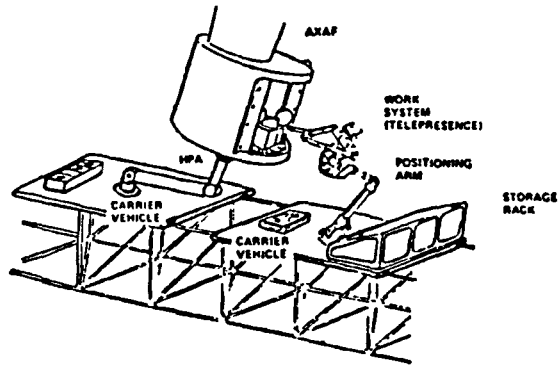


Figure 25: Telepresence Work System (TWS) by Grumman Aerospace Corp.

or on the arms themselves to provide other points of view. The ROSS sensor data is transmitted to the operator by radio link.

The ROSS does not operate automatically; an operator in shirtsleeve environment inside the space station or space shuttle remotely operates the ROSS. Special telepresence controls create the illusion that the operator is at the remote worksite: He sees through the teleoperator's eyes and feels through its arms. The images from the stereo cameras on the ROSS are shown to the operator through special optical equipment that presents the corresponding camera images to the operator's eyes. This allows the operator to "see what the ROSS sees" as a three-dimensional black and white or color image. The operator holds the handgrip of a teleoperator master control in each hand. Moving a master control moves the corresponding slave arm and vice versa, as if they were connected by stiff springs. In general, the greater the apparent stiffness, the better the operator can feel what the slave arm is doing, and the quicker he can complete a task.

The ROSS would normally be operated by the space station crew, but could also be operated from the ground. However, long-distance links would degrade performance because they introduce a time delay in the force-reflection system. When delays exceed 0.25 second, as they can with ground control, the illusion of telepresence disappears and operators tend to shift to an inefficient "move-and-wait" strategy [52].

2. *Telepresence/Robotics Experiments*

NASA has carried out many projects in telepresence and robotics [7]. The following sections describe a representative sample that are particularly relevant to the space station.

a. Orbital Servicing System (OSS)

The OSS is a fully-automatic arm mechanism for exchanging specially-designed orbital replaceable units in a satellite. Although it fulfills its function, it is a good example of a disadvantage of the "hard automation" approach: it is a special-purpose electro-mechanical device, with no capability for sensing or general-purpose computer control. It cannot be used for any purpose other than module exchange, and would therefore have to justify itself economically on that basis.

b. Protoflight Manipulator System (PFMA)

The PFMA (Figure 26) is an experimental teleoperator slave arm, designed by NASA in cooperation with Martin-Marietta [53]. Both the OSS and PFMA were built about ten years ago and are currently at NASA-Marshall. The PFMA is a prototype for one of the arms on the ROSS. It was originally designed with low-friction gearing to make the joints easily back-drivable to improve the quality

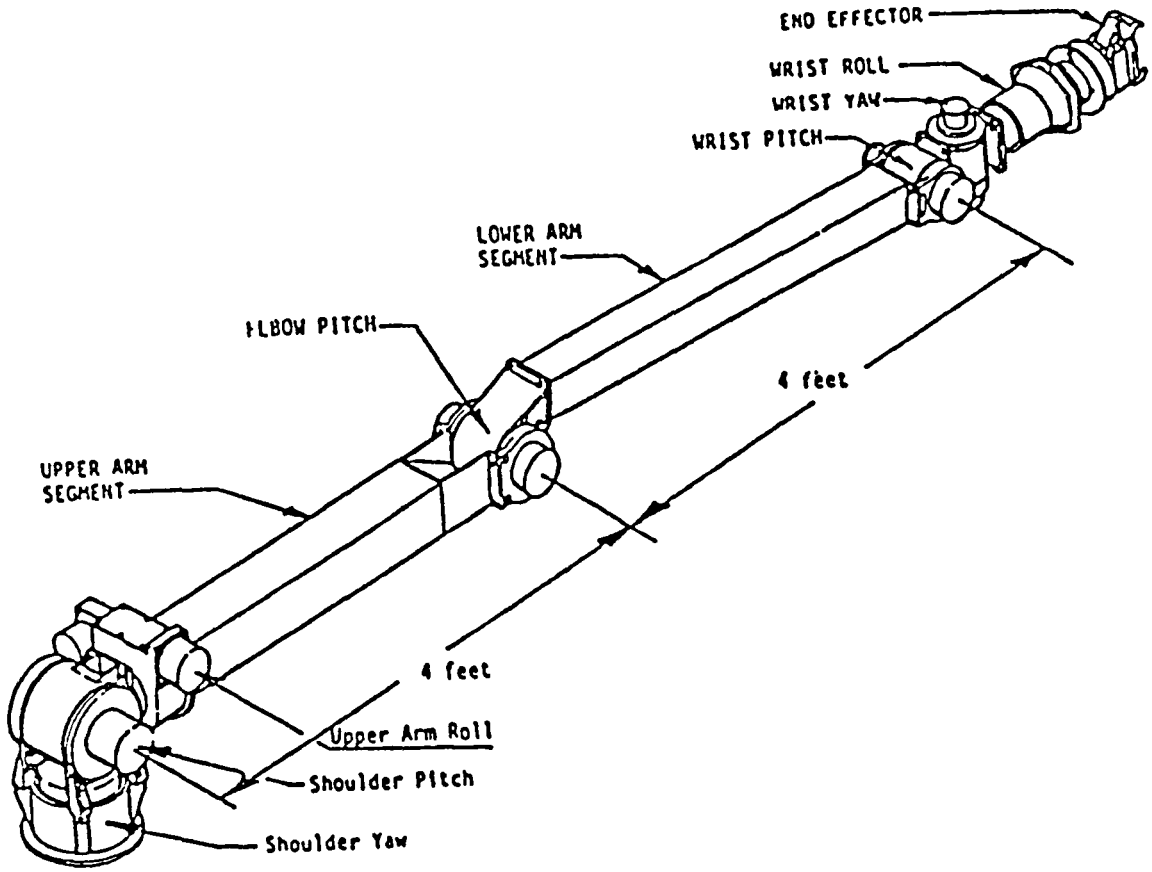


Figure 26: Proto-Flight Manipulator Arm (PFMA)

of force-reflection that could be attained. Telepresence experiments will begin in 1985 with the PFMA at NASA-Marshall, using a light-stripe three-dimensional vision system [54] and, initially, a rate-control joystick instead of master-slave control

c. MIT Beam Assembly Teleoperator (BAT)

The BAT [55] (Figure 27) is a mobile, waterproof teleoperator specifically for experiments in the neutral-buoyancy water tank at MSFC. It was built by Professor Aiken's group at the MIT Space Systems Laboratory. BAT was optimized for handling long beams in experiments on the construction of large structures in space. It has two arms—a stationary "grappling" arm designed to hold a beam stationary or move it lengthwise by means of rollers in the gripper, and a five-degree-of-freedom "dexterous" arm for connecting a second beam to the first. The BAT has a replica master control, but apparently no force-

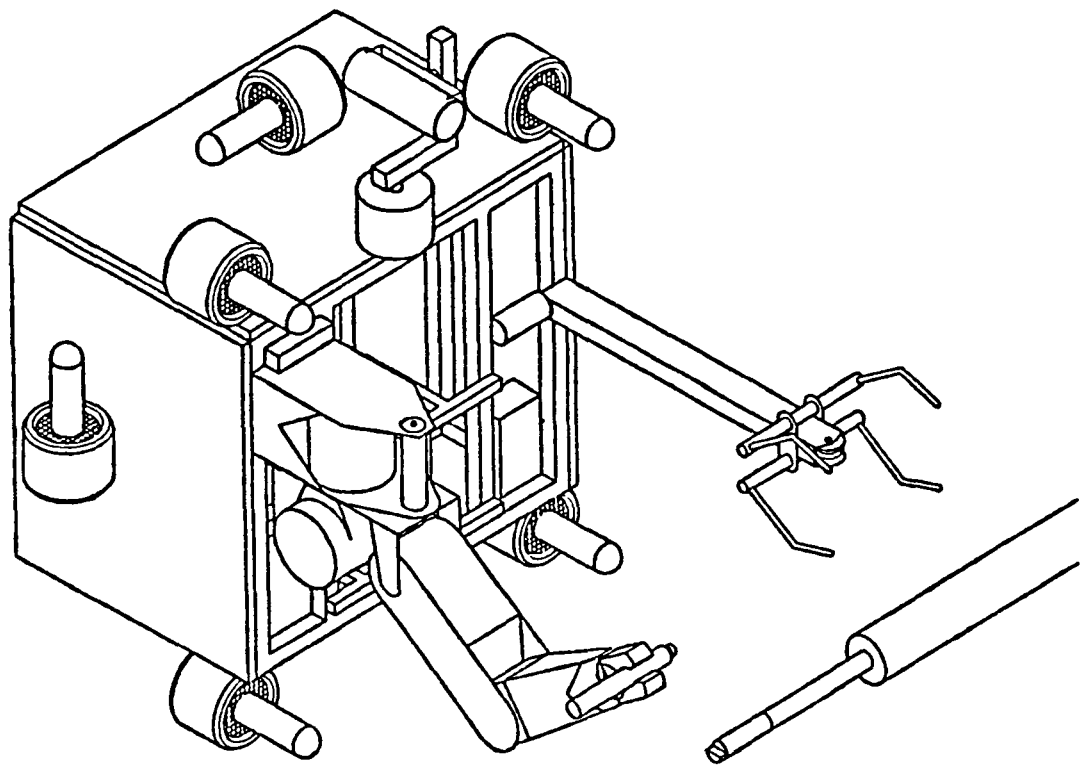


Figure 27: MIT Beam Assembly Teleoperator (BAT)

reflection capability, so the operator must rely entirely on a television picture. The BAT is a comparatively rudimentary telepresence system, highly specialized to simulate weightless vehicle dynamics in a water tank, and for the connection of one beam to another using a certain kind of mechanical coupling. In a space-qualified version, a conventional reaction control system would replace the ducted propellors. However, because the system has one arm instead of two, and because that arm has only five joints, it would probably be rather ineffective for general servicing activities compared to the ROSS or TWS.

d. Goddard High-Accuracy Assembly Robot

NASA-Goddard has developed a novel parallel-jointed robot manipulator based on the geometry of an aircraft flight simulator. It is being used as a testbed for experiments in automatic planning and execution of mechanism assembly tasks, using artificial intelligence software. The manipulator is both strong and accurate.

e. Marshall Docking Simulator

NASA-Marshall has developed a six-degree-of-freedom parallel-joint manipulator like Goddard's, except that it is larger and can move more quickly because its actuators are hydraulic rams instead of electrically-driven ballscrews. It could be used, for example, in automated docking experiments to evaluate position-determination sensors and automatic flight-control algorithms for supervisory control, adaptive robotics, and intelligent robotics.

3. *Telepresence/Robotics Facilities*

NASA research centers operate laboratories for experiments in telepresence. Some of the projects they have conducted that are significant for the space station are the following:

- The Johnson Space Center Flight Crew Integration Division, carried out a series of early teleoperation experiments twelve years ago [56]. Their test subjects (including two astronauts) compared two-dimensional black and white television to anaglyphic stereo television, and a two-armed exoskeletal controller with and without force reflection. All the test subjects decided that force-reflecting manipulators combined with stereo visual feedback were the most desirable. One of the astronauts suggested that a microphone be placed in the slave so that the remote operator could hear the motors and gears working in the slave. The sounds they made conveyed significant information about slave activity that was difficult to obtain through the visual or force-reflection channels.

- A. J. Meintel's laboratory at Langley uses two Westinghouse PUMA™ industrial manipulators for slave arms. They are equipped with wrist force/torque sensors and a parallel-jawed gripper developed by the University of Rhode Island. This laboratory investigates man-machine interface issues in teleoperation, telepresence, supervisory control, and intelligent robotics [57] for single and paired manipulators.

- W. Frost's group at MSFC will be using the PFMA in

telepresence experiments on a task board, using computer augmentation, television, and quick-change tools.

- Johnson Space Center (JSC) has a mockup of the RMS, but the RMS is probably too large for manipulation of objects on this small a scale. It could serve as a "cherry picker," however, to position a smaller-scale telepresence system with respect to a worksite.

- A. K. Bejczy's group at JPL is developing supervisory control algorithms [12] for a manipulator with force reflection. They have a JPL/CURV arm (originally designed for undersea work) and a PUMA™. They use multiple microcomputers to automatically transform motions, forces, and torques between coordinate frames in order to make the slave arm easier for a human operator to control.

Some of the advantages of JPL's methods for supervisory control include the following:

- Conventional force-reflecting master controls are kinematic replicas with the same joint arrangements, and often the same size, as the slave arms. The computer allows the master control arms to be much smaller than the slave arms. They can also have a different shape—e.g., one that is more convenient to build into a compact control station. This will make it easier to put telepresence control stations into the crowded space station, shuttle, or even small manned flyers.

- The operator may sometimes watch the work area through a camera that is to one side, mounted on a slave arm, or in some other position that does not correspond to his own view of the master controls. If the misalignment between the two viewpoints is more than about 40°, people become confused about which way to move the control arms. The computer can map motions of the master controls into motions of the slave arms so as to match the view from any particular camera, preventing operator errors and reducing fatigue.

- The computer can override the operator's motion of the control arms to prevent accidents—e.g., to keep the arms from colliding with each other or with surrounding objects, to prevent them from occluding a sensor, or to keep them out of plumes of

emissions from a satellite.

- The computer can constrain the motion of the master and slave arms to help the operator make precise motions—e.g., to move a probe along a seam of a pressurized structure to detect leaks.

- The computer can transform forces and torques to allow the operator to feel the forces and torques at an arbitrarily specified point in or around the end-effector, such as the tip of a tool.

A computer can also interpret sensory feedback as follows:

- It can overlay the television monitor image of a satellite or a structure on the space station with a line drawing showing the various parts in the scene.

- It can make accurate 3-dimensional measurements of objects in a scene.

- It can "sharpen up" a tactile image to enhance edges and reduce noise.

- It can recognize an object from its pressure pattern on a tactile array

- It can measure a small object from its pressure pattern.

a. Design of Manipulator Arms

The various manipulator arms that have been considered for space automation systems pose markedly different control problems when used in a robotic system. The region of space within which any arm can move its gripper is limited. But, to place its gripper in an arbitrary position and orientation within that region, it needs at least six joints. That number also gives the gripper enough freedom of motion to make move in any direction and to rotate around any axis. Most manipulators proposed for space use have seven joints. This extra joint provides one extra degree of freedom that may be exploited to operate the arm more dexterously—e.g., by reaching around an obstacle. *Redundant* arms (those with more than six joints) are more difficult to control than a six-jointed arm, because it is necessary to decide how to use the extra degree of freedom. An arm with fewer than six joints is also more difficult to control, but for the opposite reason: it does not have complete freedom of motion. For any given hand position, there will be some direction in which it cannot move. If this is the only direction in which a part can be installed or removed, then the arm will be

unable to complete the task. The task can be completed if the robot, as a whole, can move in that direction.

On earth, gravity often limits the range of orientations in which a part may be encountered—e.g., most manufactured parts assume one of a small number of orientations when lying on a table. Five-jointed arms can be used quite successfully in such a situation. The BAT's "dexterous" arm has only five degrees of freedom (DOF) so that there are some combinations of hand position and orientation that it cannot reach. Because objects can float freely in the zero-G space environment, they can be found in any orientation. This means that a general-purpose manipulator for space should have at least six independent DOF. Each joint usually provide a single DOF, so the arm needs at least six joints. Then it can place its hand in any position and orientation to grasp a free-floating object, as permitted by the joint motion limits.

The shuttle RMS and industrial manipulators for assembly have six DOF [58]. Such arms must assume one of a small number of different postures for each gripper position and orientation (e.g., a PUMA™ manipulator can be in any one of eight different postures). If the arm has more than six DOF, it can use the extra ones to avoid obstacles. The PFMA arm for the ROSS and the arms that Grumman has proposed for its TWS have seven degrees of freedom (DOF). These *redundant* arms can assume an infinite number of different postures for any given gripper position and orientation. A highly-redundant *tentacular* arm with many extra degrees of freedom could reach into confined spaces such as bulkheads. Such arms are well suited to space applications because, in zero gravity, their inboard joints do not have to support the weight of all the outboard joints as they do on the ground.

A dextrous servicing system simulation based on the PFMA is shown in Figure 28. The operator, wearing stereo goggles, sees a stereo display of the object being manipulated. The signals from the hand controller are processed by the computer and are sent to the PFMA servo-control electronics to actuatuate the PFMA. Figure 29 shows the end effector tool interface for the manipulator arm.

b. Design of Robot Programming Languages

An adaptive robot is controlled by a computer program. Typical computer languages for programming such robots include AL [59], AML [60], RAIL [61], and VAL [62, 63]. Such languages typically provide the following classes of built-in capability:

- Equipment operation—This includes control of any mobility equipment, such as a tracked platform, a "cherry-picker" arm, a free-flyer, or legs. It also includes any commands necessary to operate sensors, such as to make a camera take a picture [64]. A

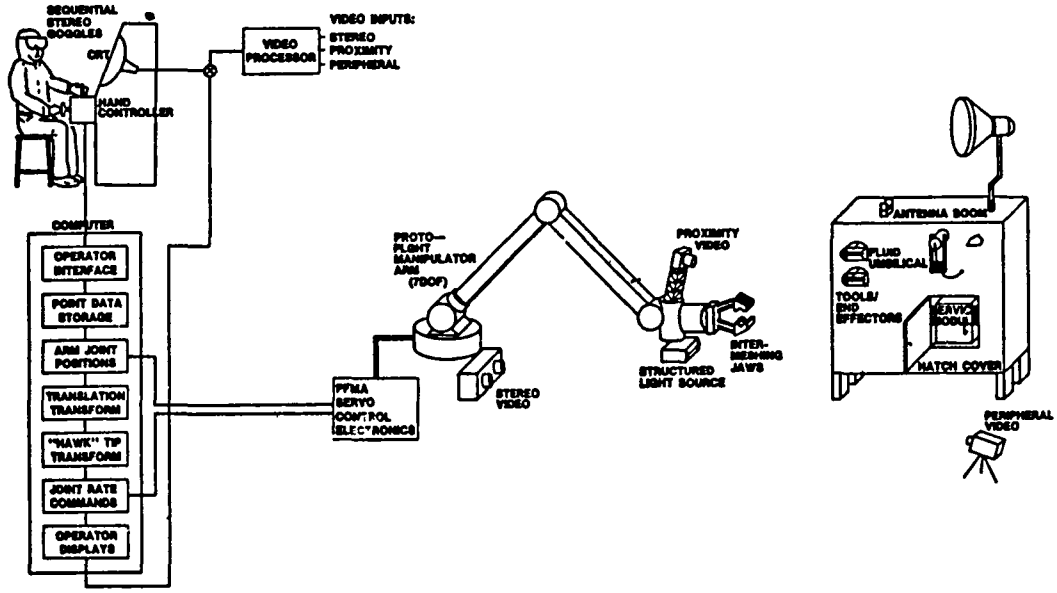


Figure 28: Dexterous Servicing System Simulation

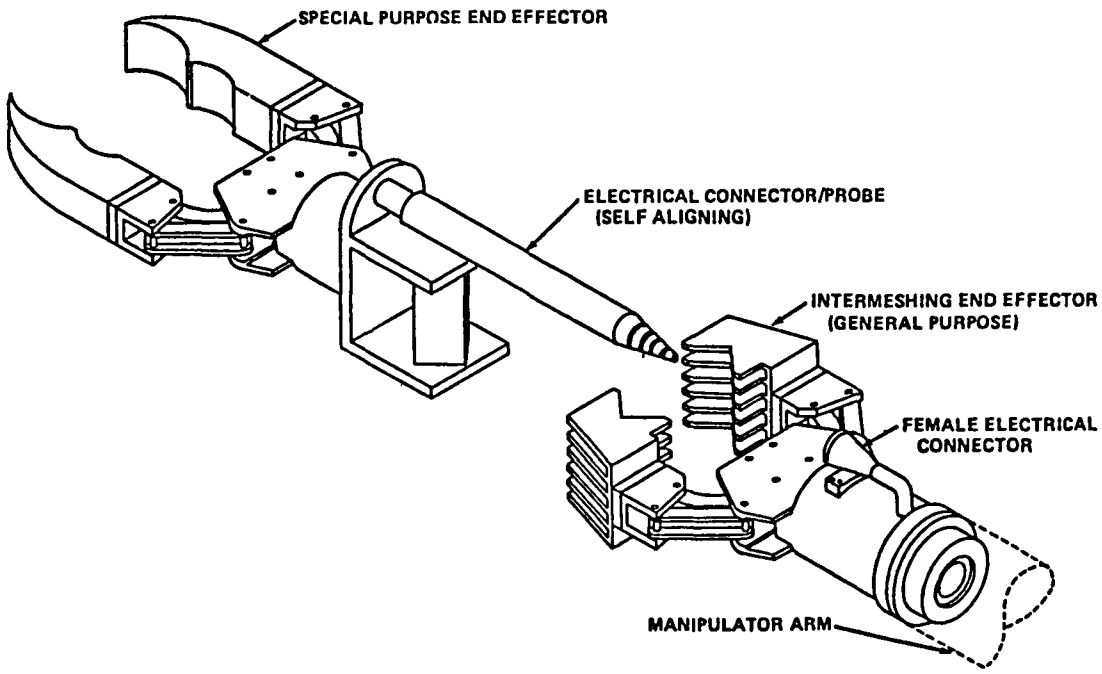


Figure 29: End Effector Tool Interface

variety of arm motions can also be provided, such as the following:

- Move the gripper to an absolute position and orientation
- Move and turn by differential amounts
- Move and turn until a force or torque limit is reached
- Move and turn at a specified speed and in a specified direction
- Follow a specified trajectory
- Move in response to information from a visual or tactile sensor

In addition, such motions may be described in terms of either joint positions or the position and orientation of the gripper expressed in a Cartesian reference frame. Several reference frames are usually available, such as a frame fixed in the base of the robot, a frame fixed in, and moving with, the gripper, frames that move in accordance with real-time sensor readings for object tracking, and user-definable frames.

- Sensory input—This may involve some routine "hidden" preprocessing, such as noise reduction. In some cases, the preprocessing may be quite elaborate, such as identifying and locating the objects in a scene viewed by a camera.

- Sequence control—This is usually provided by standard procedural programming mechanisms such as conditional branches and subroutine calls, expressed in a particular syntax.

- Computation—This should include at least integer arithmetic. However, for effective use of sensory information, it is often necessary to program analytic geometry computations, requiring a full complement of floating-point operations and mathematical functions.

- Operator communication—This generally means the exchange of alphanumeric characters through suitable computer peripherals

such as a character display and keyboard. Both simpler communication devices such as pushbuttons and more complex ones such as interactive graphic displays may also be supported.

- Communication with other equipment—This includes communication with other robots as well as with the general command and control center of the space station. An important use would be the synchronization of activities of multiple robots.

- Data base access—The simplest example is the ability to read and write data files. More advanced applications will require the ability to interface with the space station CAD data base, planning data base, etc.

Such functions may be provided by embedding them in the syntax and semantics of a "robot programming language." A simpler approach is to implement them as a set of subroutines that can be called from a conventional general-purpose programming language. Purdue University's Robot Control "C" Library (RCCL) is perhaps the most advanced example [65].

The above list indicates the fundamental or "low-level" capabilities that an adaptive robot's programming system should provide. Although such capabilities would suffice for initial experimentation, a practical programmable adaptive robot system should also provide "higher-level" capabilities to perform routine tasks. These can be implemented in terms of the lower-level capabilities. Some useful higher-level automatic capabilities for an adaptive space robot include the following:

- For a free-flyer, automatic navigation to the worksite.

- Docking and rigidization.

- Generalized "guarded moves." These are arm movements made under the supervision of a sensor, usually to stop the motion when contact occurs (or fails to occur when it should).

- Stowage and retrieval of tools and specialized end-effectors from an equipment rack.

- Placement and mating of tools with workpieces, guided by sensors and CAD models of the equipment. This may include simple cases of automatic grasping of an object with a gripper.

- Generalized compliance control at the end effector, such as

remote-center compliance for connector insertion, operation of passive mechanism such as toggle clamps, and alignment of tools with workpiece surfaces.

- Identification and location of objects by means of sensors.

E. Space Station Applications

The most important initial motivation for teleoperation/robotics automation is to enhance *crew safety* by reducing the need for extra-vehicular activities (EVAs). Another motivation is to increase *capability* by making possible some activities that are now impractical because of long communication delays, such as servicing satellites in high-radiation geosynchronous (GEO) orbits. A third motivation is to improve *productivity* of ground and space station personnel by eliminating the overhead in each EVA that is avoided,* reducing the need for constant operator attention in remote operations, and decreasing astronaut fatigue by replacing muscles with machinery. A good telepresence system may even prove to be more dexterous than a human hand in a space suit glove.

Fully-automatic adaptive and intelligent robots will increase capabilities still further [66]. They will enable continuous construction and repair where communication paths have too much time delay, are too intermittent, or are unavailable to support supervisory control—e.g., in polar, lunar, and planetary orbits. They will also increase productivity through force multiplication, since one person on the ground or on the space station can direct and supervise many robots working simultaneously on different tasks.

The concepts developed by the contractors are concerned with automating the following space station activities:

- *Manufacturing in Space (G.E.).* Use of dexterous manipulators to transfer semiconductor material between automatic-fabrication stations for the production of gallium arsenide (GaAs) integrated circuits; periodic servicing of production equipment (e.g., cleaning, replenishment of consumables); periodic rebuilding of crystal-growing furnaces.

---

* Space suit usage carries high overhead costs: the astronaut must prebreathe oxygen for an hour to avoid the "bends" due to the low suit pressure; another astronaut must be ready to egress if rescue should be necessary; it takes significant time to put a suit on, test it, take it off, and repair it; cleaning a suit after each use is a vital but time-consuming and unpleasant task. Interviewees reported a 15-to-1 ratio of overhead to productive activity.

- *Satellite Servicing (TRW).* Use of dexterous manipulators for routine servicing of satellites, such as exchanging orbital replaceable units (ORU) and refueling, as well as more difficult "Solar Max"-type repairs.

- *Construction of Large Structures in Space (Martin-Marietta).* Use of "crane" manipulators like the orbiter's remote manipulator system (RMS) to unstow modular elements (e g., beams), and to help astronauts in EVA assemble them into structures; use of such devices to transport astronauts around a structure in "cherry picker" mode.

The first two automation concepts, manufacturing and satellite service, require the most advanced equipment and control because they involve the precise manipulation of small objects. To perform this by telepresence will require good visual and tactile sensing equipment, and robots will, in addition, require very advanced capabilities in visual and touch perception, planning, and reasoning. On the other hand, the scale of orbital construction is several orders of magnitude larger than that of manufacturing or satellite service while its pace is one or two orders slower. Thus, it should not require as dexterous manipulation as either of those tasks. "Cherry picker" transport of EVA astronauts has already been demonstrated during orbiter flights.

The contractors developed the following innovative automation equipment design concepts for perception, manipulation and mobility in connection with the above functions:

Perception

- Voice control of space station automation equipment.

- Adoption of commercial telephony standards for baud rates, subcarrier frequencies, etc., to simplify the task of interfacing the space station's data management system with ground communication networks.

Manipulation

- Quick-change end effectors for servicing, manufacturing, and construction missions.

- Various kinds of workpiece positioners, such as turntables and "rotisseries."

- A modular manipulator that can be repaired by an astronaut in

EVA, or by teleoperated or robot manipulators.

- A portable, dexterous manipulator that can be attached to any of the general-purpose equipment connectors mentioned in the preceding section.

- Innovative zero-gravity tooling, such as hands that retain parts by acoustic or electrostatic levitation to avoid contaminating pure materials or damaging fragile components.

Mobility

- Transportation systems for teleoperators, robots, equipment, and people, such as rails, cables, free flyers, and "cherry pickers."

- Locomotion methods for teleoperators and robots, such as walking and a number of motion patterns variously designated as "inchworm," "push-pull," and "rotating beam."

- Mobile, manipulator-equipped, shirtsleeve-environment work stations capable of operation by a person inside, by telepresence from a remote control station, or automatically as a robot.

- Mechanisms for free-flyers to attach themselves rigidly to satellites and work areas on the space station, even where no attachment points have been installed for that purpose.

- A "double-ended" manipulator with end effectors at both ends that could "walk" from one equipment connector to another in order to reach different parts of the space station. It would be capable of "reciprocal articulation," so that either end could serve as the base of the arm. Once at the work site, it could wield a tool or gripper with its free end.

F. Enabling Capabilities for Autonomy

A number of specific capabilities must be developed to enable the contractor's automation concepts to be demonstrated with various levels of autonomy. We list these capabilities below, grouping them into three categories. The first category consists of those which will be required initially to implement the teleoperation and telepresence levels. These capabilities are mostly related to the lower, "equipment" portion of the diagram in Figure 23. The second category consists of those primarily needed for supervisory control and adaptive robotics. Most of these have to do with the middle, or "control" layer of the diagram. The

third category of AI capabilities is related to the top, or "reasoning" layer, and are needed to produce intelligent robots [67]. For each of the three categories, we further classify each technology according to the major automation function—perception, manipulation, or mobility—upon which it has the most impact.

1. *Category 1: Teleoperation and Telepresence*

Perception

- *Visual Sensing.* Displaying a television picture of the work area to the operator.

- *Proximity Sensing.* Reporting the distance to an object, usually over distances of less than a few centimeters—e.g., by shining a light on it and measuring the intensity of reflected light [68].

- *Tactile Sensing.* Communicating to the operator contact forces acting on the remote equipment. The two most important kinds of tactile sensing are the following:

  - *Force Reflection.* Transmission of forces acting on a slave manipulator back to the master control, so the operator can feel them.

  - *Tactile Array Sensing.* Reporting pressure distribution over the surface of a gripper. This might be displayed graphically, or kinesthetically through an array of tactile stimulators on the operator's fingertip.

- *Position Determination.* For guidance and navigation. External means, such as space-station-based radar, might be sufficient.

Manipulation

- *Manipulators and Controls.* A master-slave configuration with force reflection will probably provide the most dexterity. This also allows easy control over the force and torque exerted by the slave arms.

- *Grasping.* At first, a parallel-jawed gripper whose opening the operator can control. More dexterous end effectors should be provided later—e.g., with multiple articulated fingers capable of force reflection, power takeoffs, tool turrets, etc.

- *Sensor Positioning.* Primarily for positioning and pointing cameras. Tactile sensors would be carried on the gripper. A camera will probably also need provisions for remote control of focus, aperture, lens and filter selection, and assignment of camera images to display screens. The other sensors in the system will require comparatively little control, if any.

Mobility

- *Orbital Maneuvering Vehicle (OMV).* Needed to reach free-flying spacecraft, platforms, and structures. The operator should at least have the ability to fire the thrusters under remote control, using television images for guidance. However, an onboard inertial navigation system would make possible automatic piloting as well as station-keeping (maintaining a fixed distance from another object in space). This would reduce piloting skill requirements.

- *Docking/Rigidizing Equipment.* A standard NASA docking grapple like the one on the shuttle's remote manipulator system (RMS) is the simplest example. This has only two types of action: open/close the grapple wires and rigidize/unrigidize. More general docking/rigidizing equipment must be developed for the space station, so that a telepresence system can work on it anywhere, without requiring the presence of a docking probe. Special controls to operate such equipment may be unnecessary if the slave arms can deploy it and attach it to a structure.

- *In-Contact Mobility.* Requires some mechanism to move the robot system to different places on and inside the station.

2. *Category 2: Supervisory Control and Adaptive Robotics*

Perception

- *Object Location.* Locating an object by means of vision, touch, or other senses. This is easier if the object's appearance or shape is known, but the ability to detect the presence of unidentified objects would also be important.

- *Proprioception.* Sensing by a robot of the positions and motions of its own articulated structures, such as arms, fingers, legs, feet, "necks," etc.

- *Effort Sensing.* Sensing external forces and torques exerted on

its body and limbs, especially on its grippers or other end effectors.

- *Grasp Sensing.* Sensing how a gripper is grasping an object; this may involve interpretation of high-resolution tactile array images or finger-joint torques.

- *Position Determination.* (For a mobile robot.) Determining the robot's own position with respect to the space station, a satellite, etc. This could be accomplished by navigation satellites, vision, inertial guidance, or some combination of these and other means.

Manipulation

- *Task-level Control.* A method of specifying and executing the procedure (program, algorithm) to accomplish a task. This includes specifying sensor actions, computations, decisions, and communications, as well as manipulator and gripper motions. Initially adaptive robots will be programmed by specialists on the ground. Sophisticated programming aids based on CAD data bases and artificial intelligence will probably have to be developed to enable the space station crew to program robots rapidly and correctly in orbit.

- *Effector Control.* Computing each joint actuator's positions, motions, and efforts in order to position or move an articulated structure such as an arm or leg in any specified way. This usually requires kinematic computations, but these are relatively simple [69]. Rapid motion or handling of massy objects will require additional dynamic calculations, which are quite complex [70]. The control of nonrigid structures is still a research topic and probably represents the most difficult kind of control problem.

- *Coordinate Conversion.* Transforming positions, velocities, forces, torques, and other spatial quantities from one reference frame to another fast enough to support real-time control of arms, legs, cameras, etc.

- *Adaptability.* Adjusting preprogrammed motions "on the fly" to match the actual positions of objects around the robot, usually on the basis of sensory information.

- *Effort Control.* Exerting a controlled force and torque in

arbitrary directions on an object with the robot's limbs.

- *Programmable Compliance.* Adjusting the effective mechanical compliance of the robot's limbs to suit specific task requirements.

- *Gripper Control.* Controlling gripper action and grasping force to maintain a firm hold on an object without damaging it and without allowing it to slip.

Mobility

- *Long-range Navigation.* Traversing distances that are quite large relative to the size of the robot. Mainly involves position determination and course planning.

- *Short-range Navigation.* Traversing distances that are comparable to the robot's size. Mainly involves obstacle detection and avoidance.

- *Locomotion.* Operating a propulsion system to move the robot over large distances or to adjust its position with respect to objects in its work area. The propulsion system may be a conventional reaction jet (for a free flyer) or more innovative equipment, such as a rail transport mechanism, ducted fans, or legs.

3. *Category 3: Intelligent Robots*

Perception

- *Multisensory Integration.* The ability to combine information from different kinds of sensors to more correctly perceive external events and conditions. For example, visual and tactile information can complement proprioception for more precise manipulation of small objects. Combining inertial and visual sensing with navigation satellite signals would permit more accurate long-distance navigation.

- *Situation Assessment.* The ability to deduce from sensory observations and previous knowledge the important facts about its surroundings. It would have to be able to deal with incomplete or even contradictory information. Situation assessment and multisensory integration would require advanced expert-system technology.

- *Man-machine Interface.* Methodologies for communication with the space station crew. Natural language will probably be the most difficult to perfect, but the ability to speak *to* the equipment will be particularly important. It requires the smallest and simplest interface hardware, and no special training or skills to use.

Manipulation and Mobility

- *Automatic Planning.* The ability to devise a complex schedule of activities in order to accomplish a particular mission. This would involve reasoning about such factors as time constraints, resources, conflicts, plan reliability, co-ordination of multiple agents (including robots and people), and planning to obtain missing information with sensors.

- *Plan Execution and Monitoring.* Comparing the current situation to the situation anticipated in the plan, noting any problems, and taking advantages of any unplanned-for advantages that occur. This also involves deciding whether a plan has gone so far wrong that it is necessary to revise the plan.

- *Automatic Replanning.* Generating a new plan to suit the present circumstances. This is different from the original planning problem, because (1) it may have to be done rapidly, and (2) much useful information will have been generated during the original planning process that may be of use.

- *Knowledge Representation.* A crucial issue in implementing perception, as well as manipulation and mobility, for intelligent robots is the representation in the computer of many different kinds of knowledge. The representation must be *concise* so that a lot of information can be made available to a program. It must also be *convenient*, allowing easy computation or deduction of any additional information that is likely to be needed. Finally, it must be *extensible*, so that new kinds of information can be represented as new capabilities are implemented. Specific information is very important, such as how to use various tools, routine equipment operation and maintenance procedures, and the schedule of activities for people and equipment on the space station. However, an intelligent robot will also need a great deal of general, *commonsense* knowledge about topics such as time, causality, geometry, mechanisms, dynamics, and electronics to reason about and respond correctly to unexpected situations such

as multiple simultaneous equipment failures

G. Research Funding

DARPA is not interested in teleoperation or the special problems posed by a zero-gravity environment; its principal emphasis is on the development of aids to human decision-making and on automatic navigation for autonomous land-mobile robots. The DARPA Autonomous Land Vehicle (ALV) project encompasses a considerable amount of vision research dealing with terrestrial scenes, particularly roads

NASA research in telepresence that is being supported at the Office of Aeronautics and Space Technology (OAST) and carried out at the various NASA centers must be expanded and coordinated if it is to have an impact on the space station. In particular, intensive early research and development work on *telepresence* is imperative—in particular, on slave equipment hardware, work station design, and support software. A vigorous effort is necessary to provide space-qualified equipment with useful dexterity in time to be "in sync" with the contractors' schedules for implementing their automation concepts. Furthermore, any shortfalls are likely to delay NASA in attaining the high-payoff robotic capabilities that are its ultimate objective. Note that, in most of the research programs described below, we propose a set of early across-the-board "benchmark" demonstrations for approximately 1987. These demonstrations would have the following purposes: (1) to make sure that the most advanced automation technologies available are identified so they can be adapted to IOC; (2) to educate the space station community in general about these new technologies; (3) to obtain an accurate picture regarding deficiencies in performance of the various technologies, so that NASA and Congress can best direct available resources to research-and-development programs planned for 1988-1995.

H. Demonstrations

1. *Demonstration Testbeds*

The space environment differs from the earth's in having very low gravity, high vacuum, extreme glare, extremes of temperature, and occasional high levels of ionizing radiation. For innovative teleoperation and robotics systems development, it is necessary to carry out experimental testing and verification in a realistic environment. Certain demonstrations should be conducted, either separately or in combination, to show that the contractor's telepresence and robotic concepts are feasible. Some could be performed first on the ground, later in the orbiter bay, still later in "cherry picker" mode on the end of the RMS, and finally by a free flyer.

Most concepts can be adequately demonstrated initially in a 1-g environment. A few require simulated weightlessness, which can be provided by a neutral-buoyancy tank such as the one at Marshall Space Flight Center (MSFC). A "frictionless" flat floor with air bearings such as those at MSFC, Stanford, and other research centers, could also be used for some demonstrations. In all cases, proof of concept will ultimately require a demonstration in actual orbit for proper testing under conditions of weightlessness and, where appropriate, in a vacuum The planned NASA facility Space Missions for Automation and Robotic Technologies (SMART), for example, could support many of these orbital demonstrations.

SMART is a multi-flight shuttle and space station automation and robotics test facility for the evaluation of advanced robotics, automation, and telepresence technologies and real-time operational concepts. The facility, able to be flown in either the shuttle or the space station, will validate robotic and automation applications Technology and overall system capability will be upgraded periodically by the Ames Cooperative Research Team consisting of researchers from Ames, industry, and academia. The facility will be managed by NASA and open to all potential investigators, and investigators for each flight will be selected by a peer review group.

Definition and design will be carried out 1988-1987 and test and evaluation in 1988. The first SMART flight will be in late 1988. Demonstrations will be carried out in sensing, planning and decision-making, control and manipulation of flexible structures, robot control languages, and telepresence/supervisory control.

A progressive series of SMART flight demonstrations will be (1) robotic device attached to RMS, (2) teleoperated and tethered robotic system, (3) free-flying robotic system, (4) multiple free-flying robots including telepresence An example of an initial flight in 1988/89 is teleoperation with video feedback of a single arm/hand on a non-free flyer. Force and contact sensing would be used to aid in grasping in an assembly operation. An advanced flight in 1991-1993 would be a free-flyer with multiple arms, capable of assembly, and of performing servicing and fabrication. The device would have an integrated sensing capability using touch transducers, force/torque transducers, and vision/range transducers. Later, automated planning and analysis would enable the experiments to be carried out with only high level supervision on the part of the crew.

2. *Demonstration Sequence*

We recommend that each demonstration be repeated using increasing amounts of autonomy—e.g.:

(1) Telepresence

(2) Telepresence interspersed with automatic control for routine portions of a task to reduce operator fatigue or to speed up performance

(3) Adaptive robotics, using preprogrammed, sensor-controlled actions

(4) "Intelligent" robotics, using automatic planning and expert systems to decide how to carry out crew requests

The simplest of these methods, telepresence, would be sufficient to demonstrate feasibility of the contractors' concepts for IOC. Higher levels of autonomy could be demonstrated later as the enabling technologies mature. We suggest the following set of mission-related demonstrations. They are listed in approximate order of increasing difficulty within each group.

Manipulator Repair

- This demonstration should be done by astronauts in EVA, as well as by telepresence or other automated methods; its purpose would be to show that people are capable of repairing automation equipment in an emergency.

Satellite Servicing

- Exchange orbital-replaceable modules in a satellite.
- Mate and uncouple representative connectors used on spacecraft.
- Operate simple mechanisms (e.g., latches, cranks, slides, control handles).
- Transfer fluids to and from a satellite (cryogens might be featured in a separate demonstration of this type).
- Remove and install typical fasteners used on satellites (e.g., screws, bolts, nuts, clips).
- Handle nonrigid satellite materials (e.g., insulation blankets, foils, fabric, wires, hoses, springs, seals).
- Rigidly attach a telepresence/robotic system to a work area on the space station (from which the system might work on a docked satellite).

- Rigidly attach a telepresence/robotic system to a free-flying but nonrotating satellite.

- Dock with, grapple, and despin a free-flying satellite.

Manufacturing of GaAs Integrated Circuits

- Transfer wafer carriers between automatic fabrication equipment

- Service fabrication equipment

- Rebuild the crystal-growing furnace.

Orbital Construction of Large Structures in Space*

- Move along the surface of a structure being constructed. Initially simple rail or cable transport mechanisms would suffice. Free flyers and/or crawlers could be introduced later.

- Unstow structural members and convey to EVA astronauts

- Join structural members with special mechanical connectors

- Install equipment on structure (e.g., cables, lights, docking rings).

Some of these demonstrations will require progress in "conventional" technologies such as process control, as well as in robotics or artificial intelligence. For example, to transfer cryogens automatically, NASA will need zero-gravity instrumentation that can measure the quantity transferred, but none exists. One could probably identify hundreds of similar technology gaps that must be filled for the space station. However, they lie outside the scope of this report, and, for simplicity, we have assumed that the AI/robotics technologies are the pacesetters.

Figure 30 lists a schedule for demonstrating the above activities that would concur with the automation schedules proposed by the contractors.

*Construction demonstrations would necessarily involve small but representative subsections of actual structures

Ground: D --- "Dry lab" Orbit: E --- EVA
N --- Neutral T --- Telepresence
buoyancy S --- Supervisory control
tank A --- Adaptive robotics
X --- Intelligent robotics

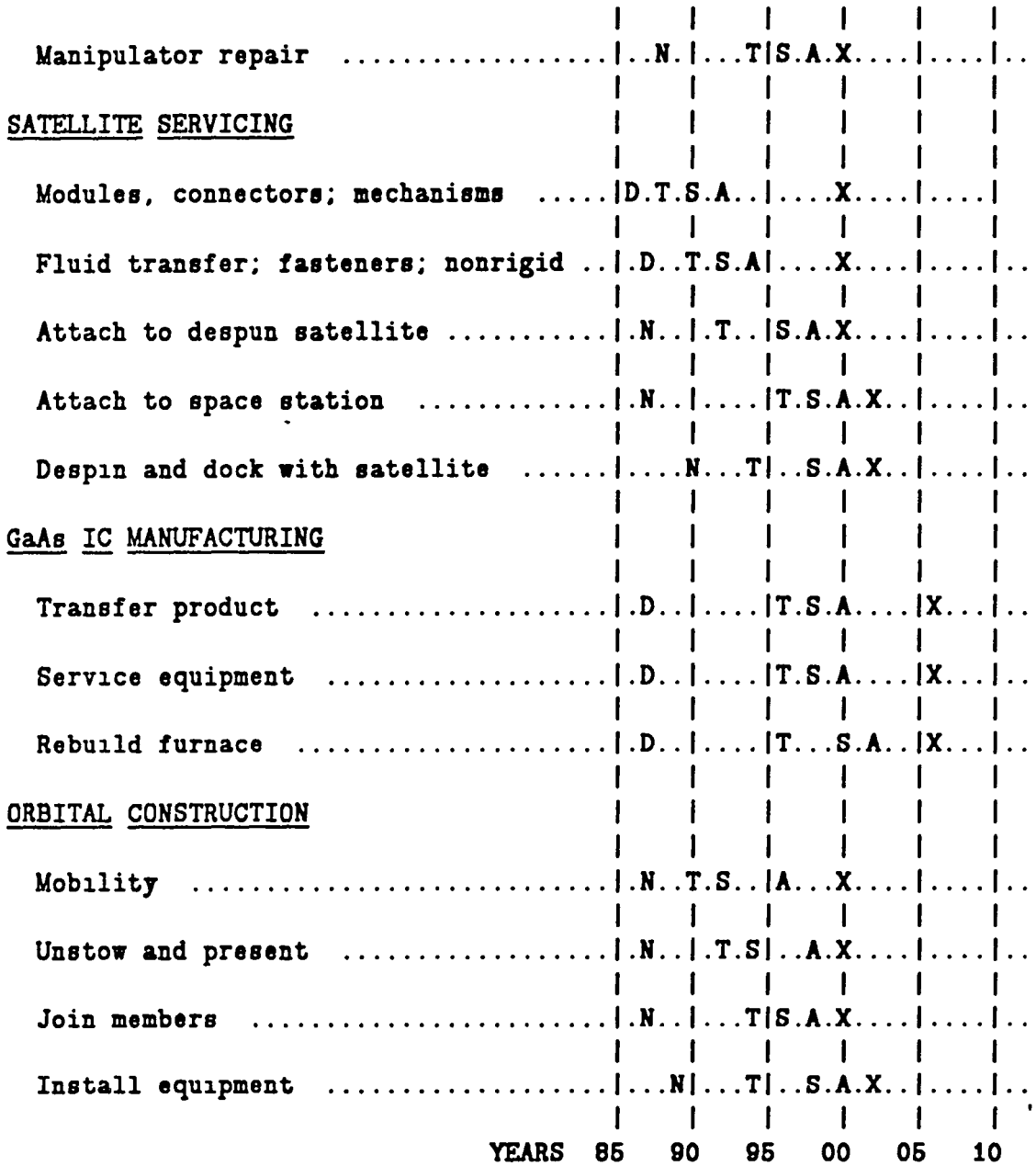


Figure 30: Schedule of Mission-Related TP/Robotics Demonstrations

I. Research and Development

NASA research in telepresence being supported at OAST and carried out at the various NASA centers must be expanded and coordinated in order to have an impact on the Space Station. In particular, intense early research and development is needed on *telepresence*—in particular, on slave equipment hardware, workstation design, and support software. A vigorous effort is necessary in order to provide space-qualified equipment with useful levels of dexterity in time to support the contractors' schedules for their automation concepts. Furthermore, any delays are likely to delay NASA in attaining the high-payoff robotic capabilities.

Figures 31 and show our suggested timelines for research related to telepresence and robotics enabling technologies.* Note that in most of these research programs we propose a set of early across-the-board "benchmark" demonstrations, around 1987 (denoted by the letter "S" in each figure). These demonstrations have the following three purposes:

(1) To make sure that the most advanced automation technologies available are identified so they can be adopted for IOC.

(2) To educate the space station community in general about these new technologies.

(3) To obtain an accurate assessment of the performance gaps in the various technologies so that NASA and Congress can better direct resources to research and development programs in the 1988-1995 time frame.

Most of these early demonstrations would probably have to take place on the ground, even though the later demonstrations of the mature technologies may take place in orbit. Some demonstrations would also necessarily be rather rudimentary because the particular technology will still be at a very early stage of development by 1987.

The research projects shown in Figures 31 and 32 are grouped into (1) ground-based telepresence experiments, (2) telepresence in orbit, (3) supervisory control in orbit, (4) adaptive robotics in orbit, and (5) intelligent robotics in orbit. The goals of these projects are given below:

---

*Sensing demonstrations are treated separately, in Chapter

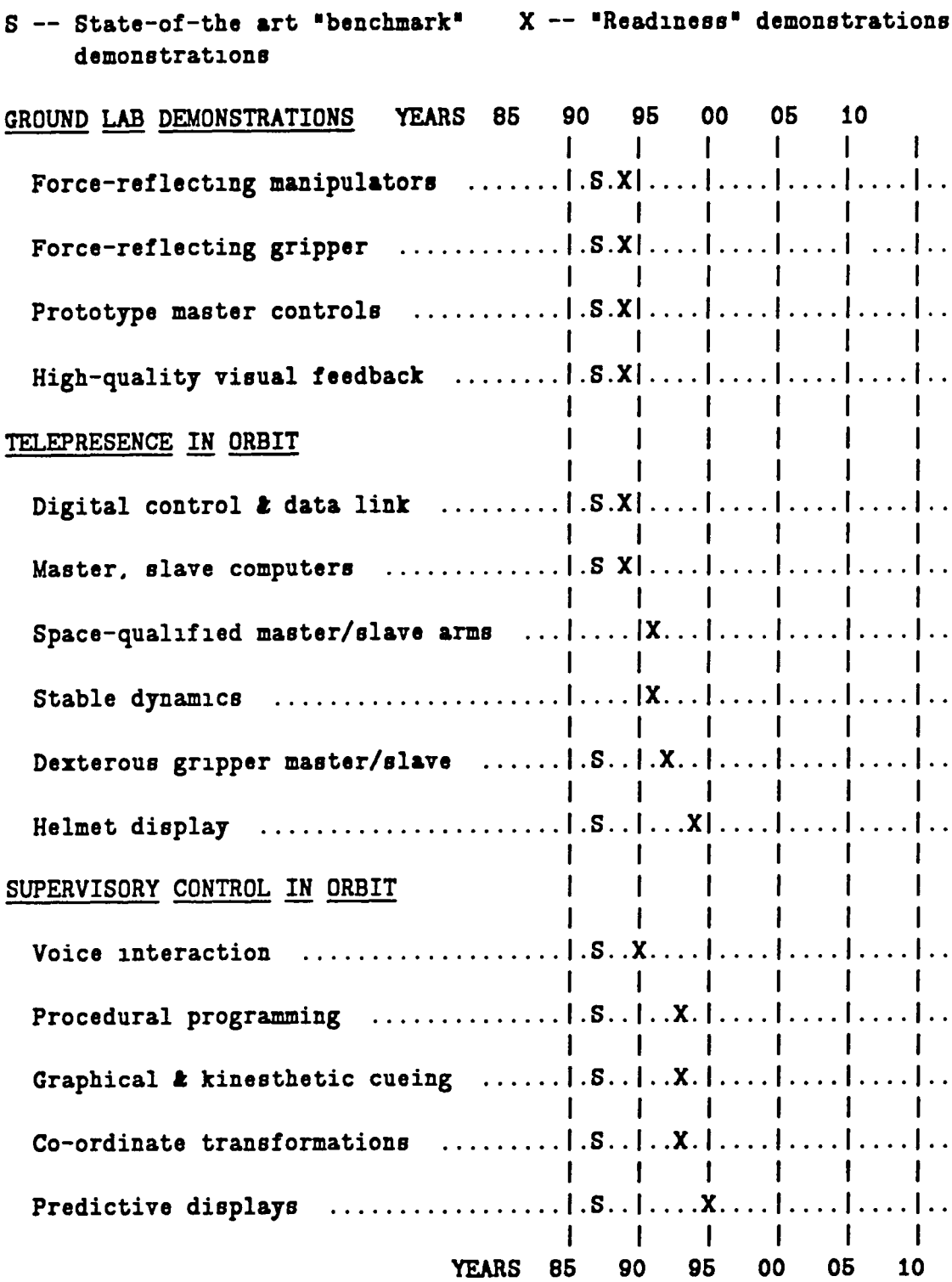


Figure 31: Research Plan for Telepresence/Robotics, Part 1

S -- State-of-the art "benchmark" demonstrations
X -- "Readiness" demonstrations

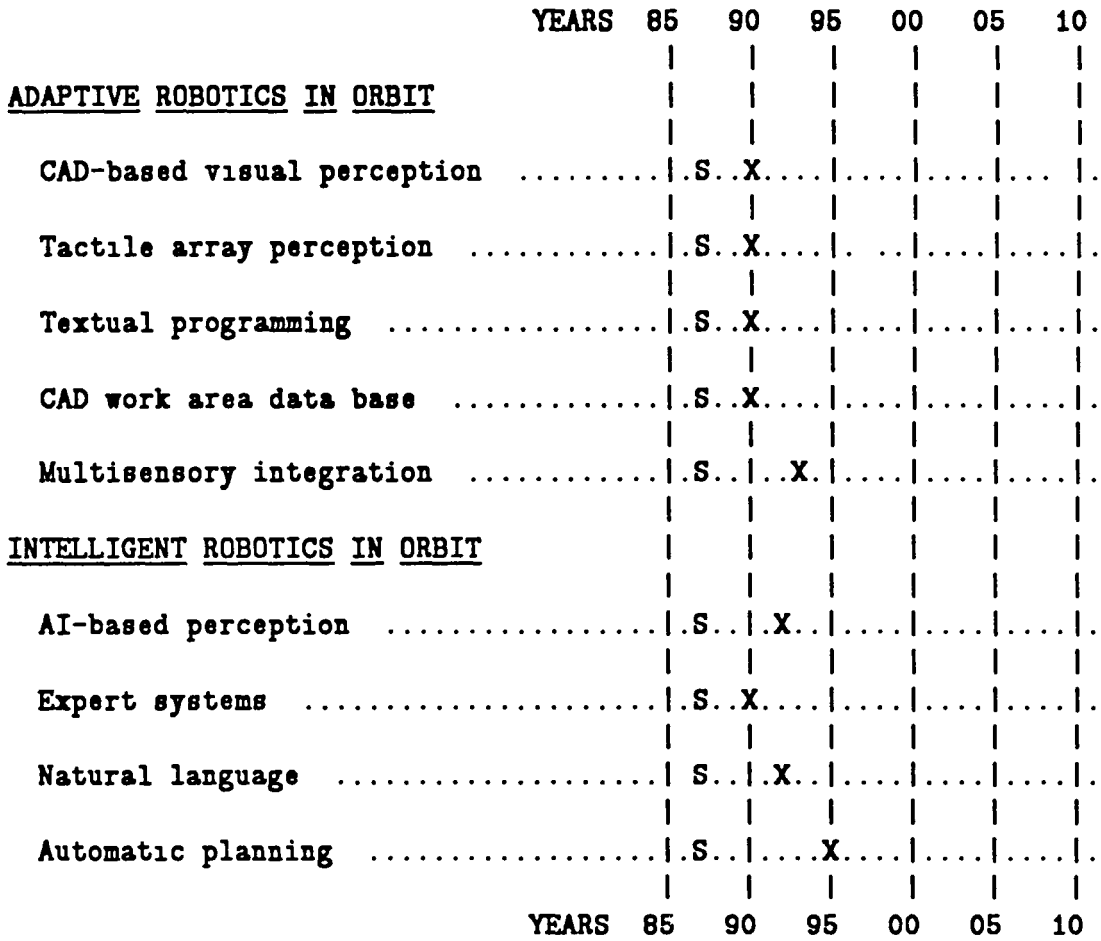


Figure 32: Research Plan for Telepresence/Robotics, Part 2

(1) Ground-based Telepresence Experiments

- *High-quality Force-Reflecting Manipulators.* A pair of teleoperated arms with force reflection of sufficient sensitivity to provide the dexterity to perform the mission-related demonstrations such as connector mating, servicing IC fabrication equipment, and joining of beams in a large structure. The arms would be designed for easy space qualification, perhaps with some modifications. This would probably require them to be electrically operated. Experiments on the ground might require some form of gravity compensation, depending on how strong the

arms are. Modularity, repairability, a quick-change end-effector socket, and other design issues should be addressed, too Force reflection might be based on measurements at the arm joints, wrist socket, finger joints, fingertip tactile sensors, or some combination.

- *Simple Force-reflecting Gripper.* A simple parallel-jawed gripper of an appropriate size and sensitivity to perform at least the simpler mission-related demonstrations.

- *Prototype Master Controls.* Controls for both slave arms and hands. These should be designed for easy space qualification for use within the habitability modules.

- *High-quality Visual Feedback.* Probably color, stereo, high-resolution television. Projected bandwidth availability for communication links will be an important consideration in choosing performance parameters. Sensors should be selected for easy upgrading to robotic use later—e.g., solid-state cameras with computer-compatible interfaces like those used in industrial robotics.

(2) Telepresence in Orbit

- *Digital Communication Link.* This is part of the general space station support facilities. It should be capable of transmitting all necessary sensory and control information between an operator and the remote equipment. Might involve packet radio techniques based on free-flying relay platforms, wide-bandwidth modulation of laser beams, or other advanced communications technology.

- *Master Computer.* A computer that operates the telepresence master control and feedback equipment. Although not strictly necessary at first, it will ease later evolution to supervisory and robotic operation if it is present in initial manually-operated equipment.

- *Slave Computer.* A computer in the slave equipment. Also not strictly necessary, but advisable for later evolution.

- *Space-qualified Slave Equipment.* Mainly, the manipulators, sensors, and transportation devices for use outside, together with any associated electronics.

- *Masters for Use in Space.* Mainly, the telepresence master controls, displays, and associated electronics for use within space station modules.

- *Stable Dynamics.* Servo control algorithms for manipulators on compliant mountings, such as cherry-picker arms and free flyers, that guarantee stability (no destructive runaway oscillations). This will probably have to include consideration of the control system for the cherry-picker or free-flyer as well as the manipulators.

- *Dexterous Gripper.* Probably a gripper with multiple multiply-jointed fingers similar to the human hand. It should be capable of the finer manipulations required to perform the more difficult mission-related demonstrations, notably in satellite service and maintenance of IC fabrication equipment.

- *Gripper Master Control.* A device that allows intuitive control of the dexterous slave gripper. Probably a replica master with its own force-reflection capability.

- *Helmet Display.* A light, compact head-up display suitable for use in or on a space suit helmet. Should be capable of displaying video from the remote cameras, overlaid with computer-generated textual or graphical information. May be a color display, and may be transparent.

(3) Supervisory Control

- *Voice Interaction.* Simple voice input equipment for controlling equipment and asking for information. Preferably, it should accept continuous speech [71].

- *Procedural Programming.* "Training" of simple automatic procedures using a combination of telepresence master control actions, voice inputs, graphic interactions, or other "intuitive" methods.

- Kinesthetic Cueing. Simulating external forces acting on the slave manipulators to help the operator move the arms more accurately. The master control computer would compute artificial tactile signals and inject them into the force-reflection channel for purposes such as the following:

- *Collision Prevention.* The master control computer simulates imaginary barriers around obstacles in the workspace. Whenever the operator moves a manipulator too close to an obstacle, the computer injects artificial signals into the force-reflection channel. Then, through the master control arm, the operator feels the arm "touch" the nonexistent barrier. The computer could prevent the slave arm from passing through the barrier. However, if such overriding of the operator's controls should be undesirable, it could instead allow him to push the arm through the barrier.

- *Motion Constraints.* The computer forces the slave arm to follow a precise path or surface in the workspace, even if the operator moves the master controls inaccurately.

- *Compliance Control.* The computer produces various useful kinds of compliance in the slave manipulators. For example, when mating connectors or reassembling equipment, it can make the slave arm exhibit remote-center compliance to prevent jamming [72].

• *Co-ordinate Transformations.* The computer solves kinematic equations rapidly in real time to map motions and forces between the master and slave reference frames. Two useful applications of this technique are the following:

  - *Reference Frame Correspondence.* Making the motions of the slave arms, as viewed by the television cameras, correspond to the motions of the master arms as viewed by the operator. This reduces operator fatigue and mistakes when the camera position does not correspond to the operator's viewpoint—e.g., when using a camera mounted to one side of the arms, or a camera carried on the gripper.

  - *Kinematic Inequivalence.* Controlling a slave arm that is a different shape from the master

control arms. This will allow the masters to be designed to suit the available space in a habitability module. It will also allow one master to operate many different kinds of slave manipulators.

- *Computer-augmented Displays.* Television displays of the work area augmented by computer-generated graphics or image processing to help a person operate the equipment. Some examples include:

  - *Graphical Cueing.* Use of information in the CAD data base to overlay a television image of complex equipment with an outline of the parts that the operator is to work on.

  - *Predictive Displays.* When there is a long time delay in the communicaion loop, the display shows a simulation of what is likely to be happening at the remote worksite before the actual report is received.

  - *Image Enhancement.* When viewing conditions are bad—e.g., because of harsh sunlight or a noisy video data link—the computer can adjust contrast or average out noise to make the image easier for the operator to interpret.

(4) Adaptive Robotics

- *Three-dimensional Imaging.* Obtaining a high-resolution "dense" range image of a scene, perhaps with brightness and color information, too [73, 74]. Would probably require an active sensor, and ideally should operate at television frame rates [75, 76, 77].

- *CAD-based Visual Perception.* Use of three-dimensional CAD models and perhaps associated information about equipment to aid location, identification, and inspection of objects in a television or range-camera image [78].

- *Tactile Arrays.* High-resolution arrays of pressure sensors, suitable for use on fingertips of manipulator hands, as well as lower-resolution "skins" with which to cover exposed parts of

slave equipment to detect contact with other objects.

- *CAD-based Tactile Perception.* Same as CAD-based visual perception, but for tactile array and force-reflection data.

- *Textual Programming Method.* A method for describing complex procedures that may involve any of the following: sensing actions, computations, decisions based on sensory data, alternate actions that are to result from those decisions, repetitive actions, simultaneous activities, externally-initiated actions, synchronous actions, access to a data base, communication with people and other automatic systems. In addition to a formal programming language, this may involve advanced man-machine interface techniques to minimize the amount of actual text that the user needs to supply, such as icons, menus, flow-charts, activity diagrams, or animated graphic simulations of the equipment in the workspace. May even involve certain kinds of expert systems for assistance.

- *CAD Work Area Data Base.* A three-dimensional geometric model of space station equipment, based on an unambiguous solid modeling technique such as constructive solid geometry (CSG) rather than an ambiguous technique such as wire frames. Should describe the relative position of each piece of equipment as well as its shape and (to an appropriate level of detail) its internal construction. This data base should be extensible to allow storage of a wide variety of other kinds of information about the equipment as well.

- *Multisensory Integration.* Integration of information from visual, tactile, navigation, and other sensors to perceive external objects and determine internal status.

(5) Intelligent Robotics

- *Expert Systems.* Use of inferential reasoning, causal models, and other techniques to draw conclusions from evidence, apply general rules to special cases, and learn from experience. Useful in diagnosis of equipment malfunction, situation assessment, and allocation of scarce resources.

- *AI-based Perception.* Use of advanced representation and reasoning techniques to interpret readings from sensors in terms of objects, events, and situations.

- *Natural Language.* Understanding what people say or type in English or other human languages, and communicating information to them in the same ways. Useful in operation of complex systems, especially vocally from a space suit.

- *Automatic Planning.* Generation of complex, detailed plans for procedures to be carried out to achieve specific results. Useful in deciding how to disassemble, repair, and reassemble equipment, how to co-ordinate repairs of a piece of automatic integrated circuit fabrication equipment with production runs of chips, and how to schedule people, equipment, and material for a large construction project in orbit.

V. TELEOPERATION AND ROBOTICS

REFERENCES

1. "SSAS: Satellite Servicing," (Final Report Z 410.1-84-160), Redondo Beach, California, TRW Space and Technology Group, November 1984. NAS 8-35081.

2. "SSAS: Automation Requirements Derived from Space Manufacturing Concepts," (Final Report NR 740-9), Valley Forge, Pennsylvania, General Electric Space Systems Division, November 1984.

3 "SSAS· Autonomous Systems and Assembly," (Final Report MCR84-1878), Denver, Colorado, Martin Marietta Aerospace, November 1984.

4. T.J. Williams, "Autonomy and Automation in Space: Its Relationship to Industrial Automation Developments," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 361-368, Las Vegas, Nevada (12-15 August 1984).

5. J.C. Fox, et al., "Study of Robotics Systems Applications to the Space Station Program," (Report KTR-108), Ann Arbor, Michigan, KMS Fusion, Inc., October 1983. NASA HQ Contract NAS W-3751.

6. C P. Fong and K. Corker, "Force/Torque Feedack Task Simulation for Advanced Remote Manipulators," *Proc., Soc. Computer Simulation* (July 1984).

7. A J. Meintel, Jr. and R.L. Larsen, "NASA Research in Teleoperation and Robotics," *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference*, San Diego, California (23-27 August 1982).

8. L. Purves, NASA Goddard Space Flight Center, personal communication.

9. T. Premack, NASA Goddard Space Flight Center, personal communication.

10. H. Plotkin, NASA Goddard Space Flight Center, personal communication, 1984.

11. J L. Adams, "An Investigation of the Effects of the Time Lag Due to Long Transmission Distances upon Remote Control, Phase II—Vehicle Experiments, Phase III—Conclusions," (Progress Report 3), Stanford, California, Department of Mechanical Engineering, Design Division, Stanford University, 1961. Final Report, NASA Contract NSG 111-61.

12. A.K. Bejczy and S. Lee, "Generalized Bilateral Control of Robot Arms," (JPL Internal Memo), Pasadena, California, Jet Propulsion Laboratories, 26 November 1984.

13. P.D. Summers and D.D. Grossman, "XPROBE: An Experimental System for Programming Robots by Example," *The International Journal of Robotics Research*, Spring 1984, *3*(1), pp. 25-39.

14. W.W. Hankins III and N.E. Orlando, "Cooperative Control: The Interface Challenge for Men and Automated Machines," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 311-318, Las Vegas, Nevada (12-15 August 1984).

15. W.L. DeRocher and R.O. Zermuehlen, "Computerized man/Machine Interface for Remote Control of a Teleoperator Arm," (Technical Support Package, Brief No. MFS-23849), Huntsville, Alabama, MSFC, 1978.

16. W.T. Park, "The SRI Robot Programming System (RPS)," *Proceeding of the 13 th International Symposium on Industrial Robots, Vol. II, Future Directions*, pp. 12-21 to 12-41, Chicago, Illinois (18-21 April 1983). Also available from the Robotics Laboratory, SRI International, Menlo Park, California.

17. C.C. Geschke, "A System for Programming and Controlling Sensor-Based Robot Manipulators," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, January 1983, *PAMI-5*(1), pp. 1-7.

18. J.W. Hill and J.C. Bliss, "Tactile Perception Studies Related to Teleoperator Systems," (Final Report 2), Menlo Park, California, SRI International, June 1971. Prepared for NASA Ames Research Center, Moffet Field, California, under NASA Contract NAS2-5409, SRI Project No. 7948.

19. J.K. Salisbury and J.J. Craig, "Articulated Hands: Force Control and Kinematic Issues," *The International Journal of Robotics Research*, Spring 1982, *1*(1), pp. 4-17.

20. C.F. Ruoff and J.K. Salisbury, "Three-Fingered Robot Hand," (Technical Support Package for Invention NPO-15959), Pasadena, California, Jet Propulsion Laboratories, Fall 1983. Also in *NASA Tech Briefs*, Vol. 8, No. 1, Fall, 1983.

21. W.E.L. Grimson and T. Lozano-Pérez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data," *The International Journal of Robotics Research*, Fall 1984, *3*(3), pp. 3-35.

22. P. Slysh, "Automated Space Structures Assembly," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 325-331, Las Vegas, Nevada (12-15 August 1984).

23. R S. Wallace, "Robotics and the Space Station," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 369-373, Las Vegas, Nevada (12-15 August 1984).

24. R.H. Cannon, Jr. and E. Schmitz, "Initial Experiments on the End-Point Control of a Flexible One-Link Robot," *The International Journal of Robotics Research*, Fall 1984, *3*(3), pp. 62-75.

25. P. Grunewald, "Car Body Painting with the Spine Spray System," *Proceedings of the 14 th International Symposium on Industrial Robots*, pp 633-642, Göthenburg, Sweden (2-4 October 1984).

26 D.E. Orin, "Supervisory Control of a Multilegged Robot," *The International Journal of Robotics Research*, Spring 1982, *1*(1), pp. 79-91.

27. M. Russel, Jr., "Odex I: The First Functionoid," *Robotics Age*, September/October 1983, pp. 12-18.

28. Park, *Control of Multilegged Vehicles*, Ph.D. Thesis, Moore School of Elec. Eng., Univ. of Pennsylvania, Philadelphia, Pennsylvania,1972.

29. M.H. Raibert, *The International Journal of Robotics Research*, Summer 1984, Vol. *3*(2). Special issue on legged locomotion.

30. M.H. Raibert, "Dynamically Stable Legged Locomotion," (Progress Report, November 1983—December 1984 CMU-LL-4-1985), Pittsburgh, Pennsylvania, Carnegie-Mellon University, 12 February 1985. Sponsored by ARPA Order No. 4184 and a grant from System Development Foundation.

31. Shao-Chi Yeh, *Locomotion of a Three-Legged Robot Over Structural Beams,* Master's thesis, Ohio State University, Columbus, Ohio,August 1981.

32. C.A. Klein and M.R. Patterson, "Computer Coordination of Limb Motion for Locomotion of a Multiple-Armed Robot for Space Assembly," *IEEE Transactions on Systems, Man, and Cybernetics*, November/December 1982, *SMC-12*(6), pp. 913-919.

33. R.A. Brooks, "Model-Based Three-Dimensional Interpretations of Two-Dimensional Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, March 1983, *PAMI-5*(2), pp. 140-150.

34. R.A. Brooks, "Planning Collision-Free Motions for Pick-and-Place Operations," *The International Journal of Robotics Research*, Winter 1983, 2(4), pp. 19-44.

35 M. Sharir and A Schorr, "On Shortest Paths in Polyhedral Spaces," (Robotics Technical Report 138), New York, N.Y., Courant Institute of Mathematical Sciences, New York University, October 1984.

36. J.K. Myers, "RCODE: The Robotic Simulator with Collision Detection," (Robotics Laboratory Technical Note), Menlo Park, California, SRI International, December 1984.

37. J.E. Hopcroft, J.T. Schwartz, and M. Sharir, "On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE hardness of the 'Warehouseman's Problem'," *The International Journal of Robotics Research*, Winter 1984, Vol. 3(4). Also available from Courant Institute of Mathematical Sciences, New York University, New York, N.Y., as Robotics Technical Report number 103, Feb 1984.

38. M.R. Cutkosky, J.M. Jourdain and P.K. Wright, "Testing and Cotnrol of a Compliant Wrist," (Robotics Institute Technical Report CMU-RI-TR-84-4), Pittsburgh, Pennsylvania, Carnegie-Mellon University, March 1984.

39. J.A. Baer and J W. Hill, "Optical-to-Tactile Image Conversion for the Blind," (Final Report), Menlo Park, California, SRI International, June 1972. Work performed for Social and Rehabilitation Service, Office of Research Demonstration and Training, SRI Project Numbers 8647 and 1417.

40. J.C. Bliss, M.H. Katcher, C.H. Rogers and R.P. Shepard, "Optical-to-Tactile Image Conversion for the Blind," *IEEE Transactions on Man-Machine Systems*, March 1970, *MMS-11*, pp. 58-64.

41. A.K. Bejczy, R.S. Dotson and H.C. Primus, "Displaying Force and Torque of a Manipulator," (Technical Support Package for Invention Report NPO-15942/5394), Pasadena, California, Jet Propulsion Laboratories, Winter 1983 Also in *Nasa Tech Briefs*, Vol. 8, No. 2, Winter 1983, item No. 18.

42. A.K. Bejczy, R.S. Dotson, J.W. Brown, and J.L. Lewis, "Manual Control of Manipulator Forces and Torques Using Graphic Display," *Proceedings, 1982 IEEE Int. Conf. on Cybernetics and Society*, Seattle, Washington (28-30 October 1982).

43. A.K. Bejczy and R.S. Dotson, "A Force-Torque Sensing and Display System for Large Robot Arms," *Proc., IEEE Southeastcon '82*, Destin, Florida (4-7 April 1982).

44. A.K. Bejczy, R.S. Dotson, J.W. Brown, and J.L. Lewis, "Force-Torque Control Experiments with the Simulated Space Shuttle Manipulator in Manual Control Mode," *Proceedings of the 18 th Annual Conference on Manual Control*, Dayton, Ohio (8-10 June 1982).

45. "Remotely Operated Gripper Tracks Applied Force," (Technical Support Package MSC-20241), Houston, Texas, NASA Lyndon B. Johnson Spaceflight Center, Spring 1984. Also in *NASA Tech Briefs*, Vol. 8, No. 3.

46. "Concept Design of the Payload Handling Manipulator System," (Tech. Rep. JSC-09709), Houston, Texas, NASA Johnson Space Center, June 1975. NASA Task No. 504.

47. R.T. Schappel, R. Spencer, et al., "Final Report, Technical Volume, Conceptual Design Study, Remote Orbital Servicing System (ROSS), Volume II," (Tech. Rep. MCR-82-533), Denver, Colorado, Martin-Marietta Corp., April 1982 sponsored by NASA Contract NAS1-16759, Task 8, Contractor's Technical Monitor: J. Pennington.

48. A.J. Meintel, Jr. and R.T. Schappell, *Remote Orbital Servicing System Concept.* paper presented at the Satellite Services Workshop, 22-24 June, 1982, Johnson Space Center, Houston Texas.

49. L M. Jenkins and R. Olson, "Remote Operating Systems for Space-Based Servicing," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 306-310, Las Vegas, Nevada (12-15 August 1984).

50. R. Olsen, et al., "Final Briefing, Analysis of Remote Operation Systems for Space-Based Servicing Operations," (Tech. Rep. SA-ROS-RP-10), Bethpage, New York, Grumman Aerospace Corp., 26 November 1984. Prepared for NASA Johnson Space Center under NASA Contract NAS9-17066, Contracting Officer's Representative: Lyle Jenkins.

51. J.E. Pennington, "A Rate-Conrolled Teleoperator Task With Simulated Transport Delays," (Technical Memorandum 85653), Hampton, Virginia, NASA Langley Research Center, 1983.

52. M.F. Fagg, N.S. Shields, Jr., R.C. Rodriguez, and J.W. Halsam, Jr., "Protoflight Manipulator Assembly," (Final Report H-84-01, H-84-02, and H-84-03), Huntsville, Alabama, Essex Corp., April 1984. NASA Contract NAS8-35320.

53. J.S. Albus, "Final Report: Proximity-Vision System for Protoflight Manipulator Arm," (Final Report NBSIR 78-1576), Washington, D.C., National Bureau of Standards, August 1978. Prepared under MSFC Order No. H-300938.

54 E B. Shain, et al., *Design and Control of A Beam Assembly Teleoperator,* Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts,May 1983. Sponsored by NASA Contract NAGW-21. Also available as Technical Report SSL#24-83.

55. R.L. Harvey, "Study of Performance of EVA Tasks with Manipulator Systems," (Tech. Rep.), Houston, Texas, NASA Lyndon B. Johnson Spaceflight Center, May 1973.

56. N.E. Orlando, "A System for Intelligent Teleoperation Research," *Proceedings, AIAA Computers in Aerospace IV Conference,* Hartford, Connecticut (25 October 1983).

57. L.L. Toepperwein, M.T. Blackmon and W.T. Park, *Robotics Applications for Industry, A Practical Guide,* Noyes Publications, Park Ridge, New Jersey, 1983.

58. S M. Mujtaba, "Current Status of the AL Manipulator programming System," *Proceedings of the 10 th International Symposium on Industrial Robots*, pp. 118-127, Milan, Italy (1980).

59. R.H. Taylor, P.D. Summers, and J.M. Meyer, "AML: A Manufacturing Language," *The International Journal of Robotics Research*, Fall 1982, *1*(3), pp. 19-41.

60. Automatix, Inc., *RAIL™ Reference Manual ER-CB-02, Autovision™ Rev 1.0,* Burlington, Massachusetts, Author, January 1981.

61. B.E. Shimano, "VAL—A Versatile Robot Programming and Control System," *Proceedings of the 3 rd International Computer Conference*, pp. 878-883, Chicago, Illinois (November 1979).

62. B.E. Shimano, "VAL—II: A New Robot Programming System for Automatic Manufacturing," *Proceedings of the International Conference on Robotics*, pp. 278-292, Atlanta, Georgia (13-15 March 1984).

63. R.H. Taylor, P.D. Summers, and J.M. Meyer, "AML/V: An Industrial Machine Vision Programming System," *The International Journal of Robotics Research*, Summer 1982, *1*(3), pp. 42-56.

64. R.P. Paul, "Introduction to RCCL: A Robot Control "C" Library," *Proceedings of the 1984 International Conference on Robotics*, pp. 293-297, Atlanta, Georgia (March 1984).

65. A.K. Bejczy and K. Corker, "Robotics and the Space Station," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 319-324, Las Vegas, Nevada (12-15 August 1984).

66. W.W. Hankins III, J.E. Pennington, and L.K. Barker, "Decision-making and Problem Solving Methods in Automation Technology," (Technical Memorandum 83216), Hampton, Virginia, NASA Langley Research Center, May 1983.

67. C. Wampler, "Multiprocessor Control of a Telemanipulator with Optical Proximity Sensors," *The International Journal of Robotics Research*, Spring 1984, *3*(1), pp. 40-50.

68. J.M Hollerbach and G. Sahar, "Wrist-Partitioned, Inverse Kinematic Accelerations and Manipulator Dynamics," *The International Journal of Robotics Research*, Winter 1983, *2*(4), pp. 61-76.

69. W.J. Book, "Recursive Lagrangian Dynamics of Flexible Manipulator Arms," *The International Journal of Robotics Research*, Fall 1984, *3*(3), pp. 87-101.

70. A.K. Bejczy, R.S. Dotson, J.W. Brown, and J.L. Lewis, "Voice Control of the Space Shuttle Video System," *Proceedings, 17 th Annual Conf on Manual Control*, Los Angeles, California (16-18 June 1981).

71. T. Lozano-Pérez, M.T. Mason, and R.H. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots," *The International Journal of Robotics Research*, Spring 1984, *3*(1), pp. 3-24.

72. R.O. Duda and D. Nitzan, "Low-Level Processing of Registered Intensity and Range Data," *Proceedings of the 3 rd International Joint Conference on Pattern Recognition*, pp. 598-601, Coronado, California (November 1976).

73. D. Nitzan, A E. Brain, and R.O. Duda, "The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, February 1977, *65*, pp 206-220.

74. R.A Jarvis, "A Perspective on Range Finding Techniques for Computer Vision," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, March 1983, *PAMI-5*(2), pp. 122-139.

75. J.M. McLauchlan, W.C. Goss, and E.F. Tubbs, "SHAPES: A Spatial, High-Accuracy Position-Encoding Sensor, for Space-System Control Applications," *Proceedings of the 1982 Annual Rocky Mountain Guidance and control Conference*, Keystone, Colorado (30 January—3 February 1982). Available as Report AAS 82-032.

76. J.M. McLauchlan, et al., "Laser Rangefinder and Remote Profilometer," *NASA Tech Briefs*, Summer 1984, *8*(4), pp. 482-483.

77. R.C. Bolles, *3DPO: A Three-Dimensional Part Orientation System.* To appear in *The International Journal of Robotics Research.*

VI SENSORS

A. Introduction

The sensing process consists of converting the relevant object properties into a signal, then transforming this signal into the information required to plan and execute a robotic function. Processing is often divided into *preprocessing* (improving the signal), and *interpreting* (analyzing the improved signal and extracting the required information). Various sensing modes—visual, tactile, acoustic, etc.— can be employed to suit different situations, and information from different sensors can be combined for a more comprehensive situational assessment.

Some model of the operating environment and its relation to the sensor is necessary for any evaluative analysis. The more autonomous a robotic system is, the more difficult the assessments it must make and the more elaborate the models it requires.

B. State of the Art

Visual, distance, and force sensors produce signals that are converted to an array of numbers, as shown in Figure 33. The array is then analyzed to obtain an understanding of the environment. We describe below the state of the art in each type of sensor.

1. *Vision*

Vision is the most useful sense a robot can possess, since it enables the robot to rapidly identify, inspect, and determine the position of distant objects without the need to touch them. Today's solid-state television cameras can operate on either visible or infrared light. The highest image resolution available (800 x 800 pixels) is now about twice that of broadcast television, and the fastest cameras can take 2,000 pictures per second (as compared with 30 for broadcasting). Nondestructive-readout cameras can store an image for hours; moreover, it can also be modified by a computer while it is in storage. Resolutions of 2000 x 2000 picture elements should be available within the decade.

The process of automatic visual perception must deal with the problem of analyzing and interpreting signals from the image sensor. Shadows, reflections, texture, and occluded parts are some of the image degradations that make it difficult for a vision system to "understand" what is being sensed. Stored

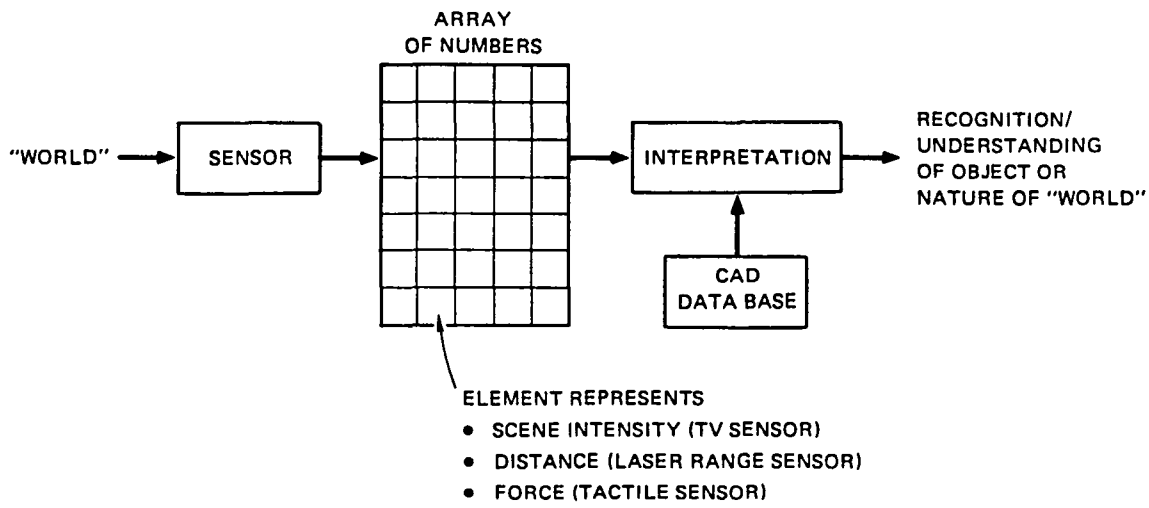


Figure 33: The Sensor Interpretation Problem

geometric models of objects being sensed can be employed for more effective analysis Figure 34 shows how a CAD data base is used in conjunction with edge analysis of an image to obtain an interpretation of the scene.

A good overview of computer vision is given in Gevarter [1]. He anticipates that 25% of all industrial robots will be equipped with some form of vision system by 1990. Rosenfeld's review of computer vision research for industrial applications [2], states that a major increase can be expected in the industrial applications of computer vision, but that significant improvements will be necessary in many of the basic techniques employed in computer vision. Extensive work is needed on developing models for classes of scenes, at levels ranging from statistics (of variations in surface reflectivity, orientation, etc.) to the nature of objects and their relationships in the scene. Systems have been developed that successfully perform many types of specialized visual tasks, but much remains to be done before we attain general-purpose vision systems with capabilities analogous to those of animals or humans. The general problem of sensor interpretation and integration is discussed in a later section.

2. *Tactile Sensors*

Tactile sensors either detect when the robot hand touches something or measure some combination of force and torque components that the hand is exerting on an object. The term *tactile sensor* means the continuous-variable sensing of forces in an array, as contrasted with *simple touch*, i.e., simple binary sensing at a single point. Tactile sensing implies skinlike properties, force-

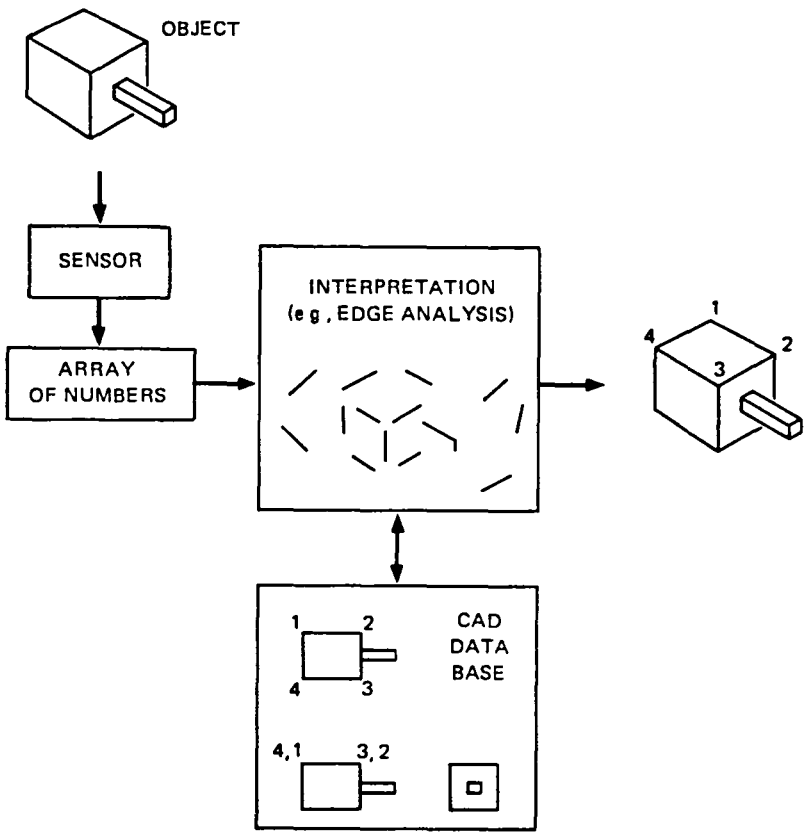


Figure 34: The Role of the CAD Data Base in Interpretation

sensitive surfaces that are capable of measuring pressure and shear forces. The tactile sensor can indicate the nature of the object being grasped; this includes the object's resiliency, its surface texture, surface normal, bounding outlines, surface curvature, and shape. Although existing commercial tactile sensors are quite simple, having only a few touch elements, promising research is leading to such devices within five years that will have the following specifications [3] 10x10 elements in 1" square; sensitivity of 1 gram with an upper limit of 1000 grams; low hysteresis, response time of 1 ms; a robust skin,

Until recently, an N x N sensor array required that N^2 output leads be sent to the processing and analysis device. The newer approaches, however, use some form of multiplexing to decrease the number of wires. In Raibert's tactile sensor [4], each sensor element is provided with its own computer, and the processed signal is shifted across rows to obtain the output. Some interesting

tactile sensors recently reported are the following

Conductive rubber. This sensor, shown in Figure 35 [5], is based on an old concept, that of a conductive material whose resistance changes with pressure. Each element can sense forces measuring 1 to 100 grams, and 256 elements fit "on a fingertip." The device has 32 output wires; scanning of the sensor array is accomplished by applying a voltage to one column at a time and noting the row responses.

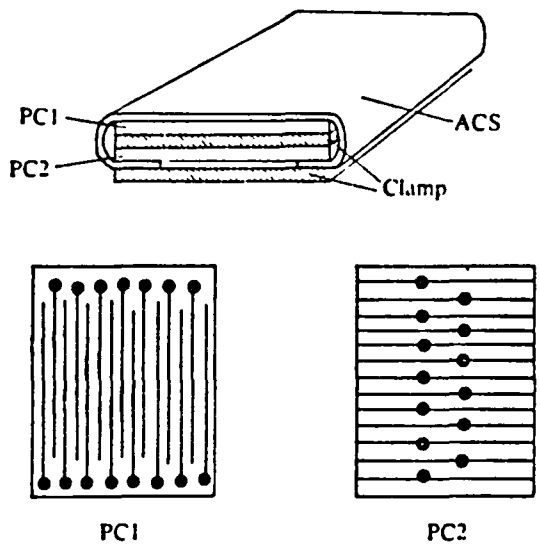


Figure 35: Conductive-Rubber Tactile sensor

Magnetic dipole. This sensor, shown in Figure 36, is a novel approach that uses a magnetic dipole embedded in a compliant medium [6]. Movement of the dipole is sensed by a magnetoresistive element. Because the dipole can move in three directions, it is possible to sense torques. A 7 x 7 array has been built that uses 2 x 2 mm elements.

Tactile-sensing computer. This approach, shown in Figure 37, employs a pressure-sensitive rubber interacting with a VLSI active circuit. This sensor/processor combination is important because it is indicative of future sensors that will use an individual processor for each element to carry out the

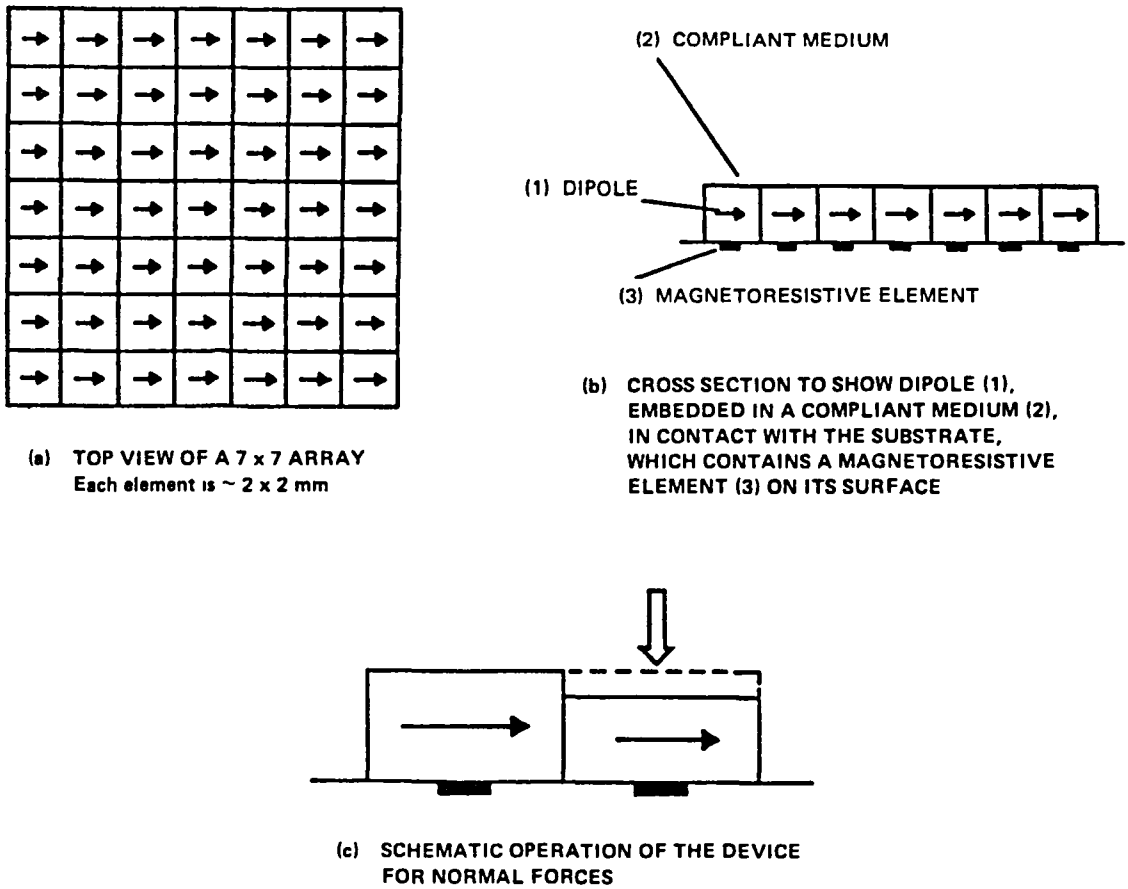


Figure 36: Magnetic-Dipole Tactile sensor

preprocessing operations. Each processor receives information from its neighbors, so that various filtering or enhancement procedures can be computed. A 6 x 3 array having 1 x 1-mm cells has been reported [4].

3. *Proximity Sensors*

Range sensors are an important means of determining the location of objects with respect to the robot. Acoustic range sensors (suitable for use only within the pressurized environments of the space station) are accurate to about one millimeter over several meters. Laser range finders are accurate to about one meter over a kilometer; with a retroreflector on the target, however, they can be made precise to about a millimeter over the same extent. A scanning laser rangefinder has been developed that simultaneously measures the reflectance of an object as well as its distance. This produces precisely registered range and intensity images.

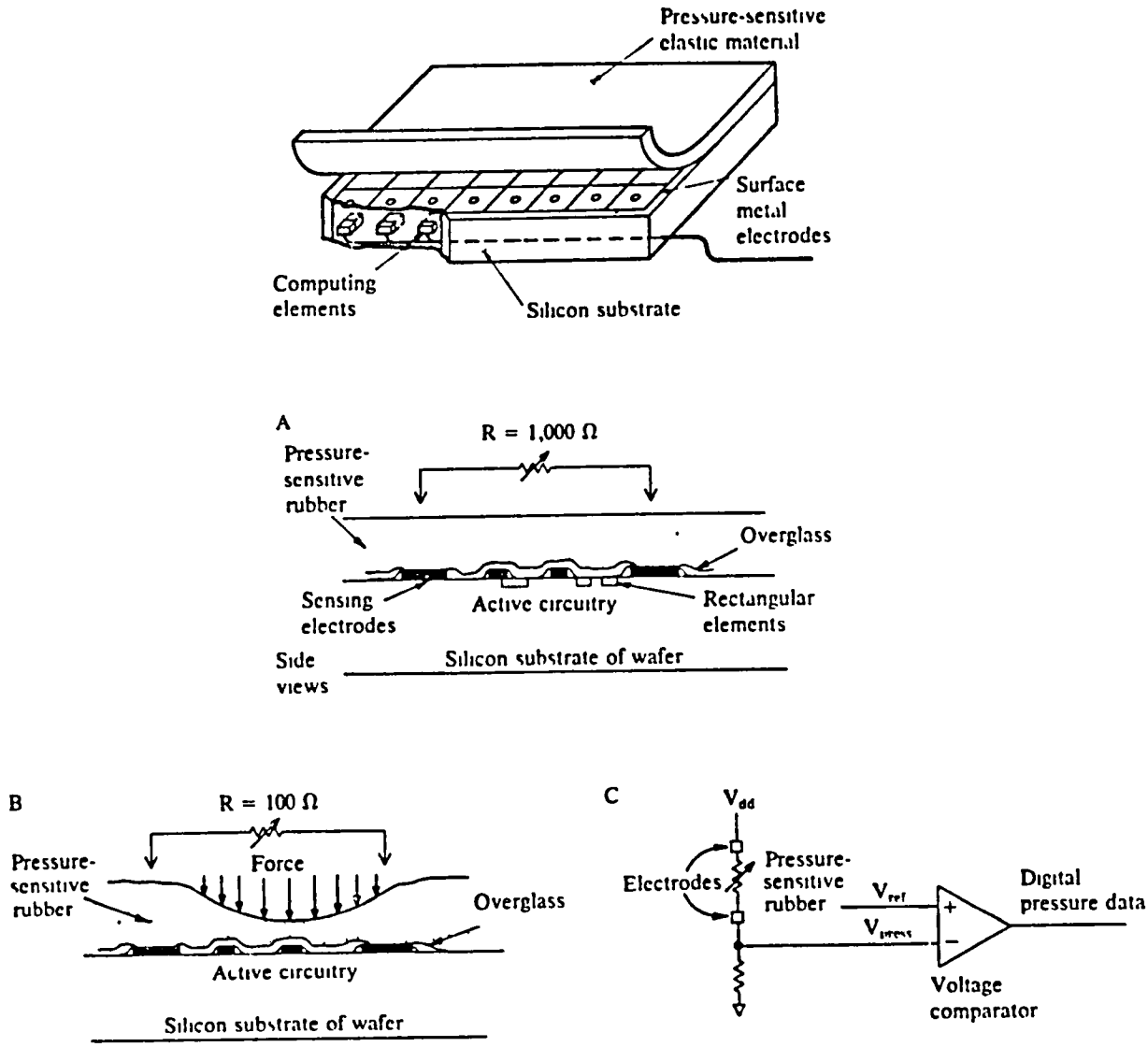


Figure 37: Tactile Sensing Computer

The main drawback to current range finders is that they must be scanned slowly over a scene to obtain the range data. Also, the transverse resolution (beamwidth) of acoustic rangers and the range resolution of laser rangers may be too coarse to be useful in many manipulation tasks.

Electro-optical devices that operate in picoseconds are now being developed. These will improve the resolution of laser rangers to the millimeter range without

the need for a retroreflector on the target object. Three emerging technologies promise significantly faster processing of range data and images than is possible with present-day electronic silicon devices. These are gallium arsenide, all-optical transistors, and Josephson junctions.

Projections of the state of the art in proximity sensors [3], indicate a range of 1-2 meters, with a resolution of 1 mm at 1 meter. A sensing and data processing time of only 10 ms can be expected.

An early proximity sensor is the Nation Bureau of Standards Proximity-Vision System [7] consisting of two separate but complementary subsystems: (1) A solid-state TV camera (128x128 pixels) mounted on the manipulator wrist; (2) coordinated with this camera is a high-intensity strobe flash system with optics that projects a thin fan-shaped plane of light into the region viewed by the camera (Figure 38). Another approach is to use a pair of close-range infra-red proximity sensors mounted in the fingertips, (Figure 39).

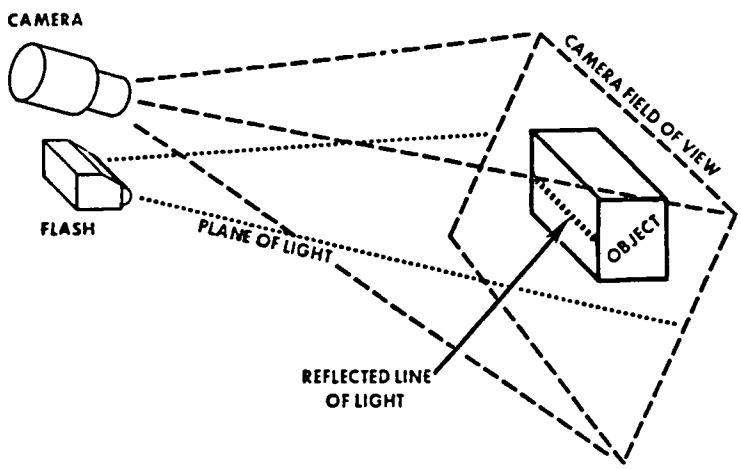


Figure 38: Plane of Light Used to Obtain 3-D Characteristics of Objects

A more recent proximity sensor for distances of 10 to 50 mm has been reported [8]. As shown in Figure 40, a light beam is transmitted to the object and its return is constrained by a pin-hole. A linear array of photosensors is used to detect the return and the distance to the object is determined using simple geometry.

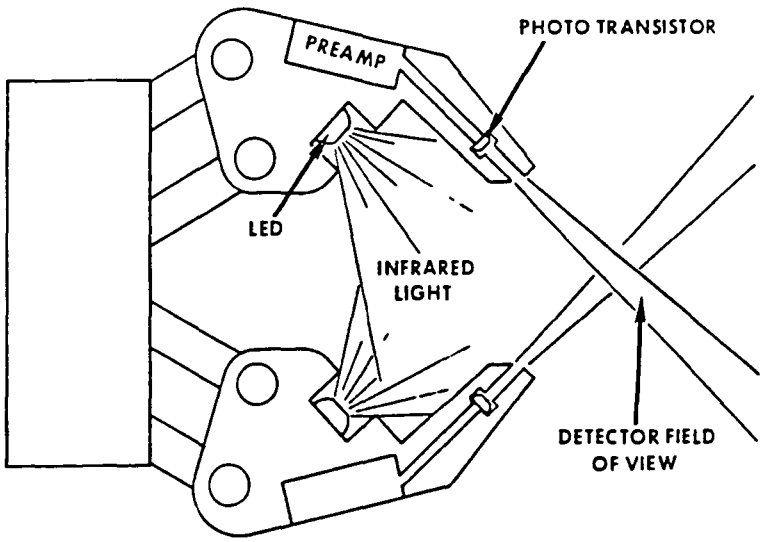
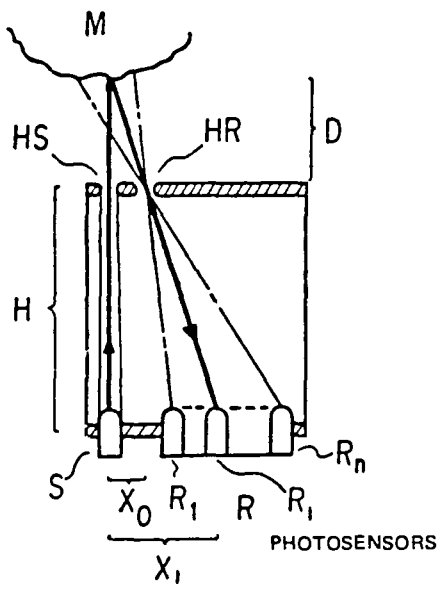


Figure 39: Proximity Sensors



M SURFACE OF AN OBJECT

S A LIGHT SOURCE WHOSE BEAM DIVERGENCE IS SMALL

R AN ARRAY OF PHOTOSENSORS

D DISTANCE BETWEEN THE OBJECT AND THE FRONT LINE OF THE SENSOR HEAD

H INTERVAL BETWEEN THE FRONT LINE AND THE BACK LINE ON WHICH *S* AND *R* ARE LOCATED

HS A PINHOLE FOR LEADING THE LIGHT FROM *S* TO *M*

HR A PINHOLE FOR GUIDING DIFFUSELY REFLECTED LIGHT FROM *M* TO *R*

X_0 INTERVAL BETWEEN THE TWO CENTERS OF *HS* AND *HR*

X_1 INTERVAL BETWEEN *S* AND *R*

Figure 40: Light Beam Proximity Sensor

4. *Proprioception*

Proprioception in robotics is the sensing of the pose of a mechanical manipulator, leg, or other jointed mechanism. This is used mainly in two ways in controlling the mechanism whose pose is sensed; in sensing the pose of a teleoperator master arm so as to command the motion of a slave arm.

Proprioception involves measuring the angle of each rotary joint and the extension of each telescoping joint in a mechanism. The joint position sensors are usually potentiometers, resolvers, or encoders. Today joint position sensors are accurate enough to enable a six-joint manipulator to place its hand anywhere within a three-meter-radius working volume with an accuracy of one-millimeter. Highly accurate sensors for joint angles or extensions are delicate, expensive, and difficult to manufacture. They are also too large for use in miniaturized robots. In the future, it may prove easier to measure the position of the hand directly than to infer it from precise measurements of each joint position.

5 *Sensor Interpretation*

We briefly discuss the problems involved in interpreting the signals received from sensors and in trying to integrate sensor information. Most of the effort in interpretation has been devoted to computer understanding of visual images.

a. Computational Vision

The general goal of *computational vision* is to develop mechanisms for interpreting visual images, i.e., to convert a video (or other) signal to a symbolic description for the purpose of identifying or locating objects, detecting changes or defects, or describing a general scene. The same image may in fact have many descriptions, depending on the reasons for processing it. One goal may be to count all the objects in an area, another may be to describe them, another may be to determine their exact location (without identifying them), and another to find defects on them. Interpretation is usually divided into *low-level, intermediate-level,* and *high-level* analysis.

Low-level vision extracts local data without the use of more general types of knowledge. This includes the detection of edges and the delineation of regions in the scene.

Intermediate-level vision deduces the three-dimensional shape of objects from the images, using shadows, textures, and edges as the clues.

High-level vision combines knowledge about objects (shape, size, relationships), expectations about the image (what might be in it), and the purpose of the processing (identifying objects, detecting changes) to aid in interpreting the image.

Progress in the field of computational vision in these three categories is shown in Figure 41. The main research topics indicated for the present era are (1) a computational theory of shape recovery, (2) model-based vision systems, and (3) applying vision systems to commercial applications.

Although vision systems are becoming available, there are many remaining research problems. Basic research in computational vision is devoted to understanding how further knowledge and reasoning can be used to interpret images, particularly so-called natural scenes, such as those found outdoors, where there are no restrictions on the environment, the objects, or the lighting. Some of the research problems currently being explored include

- Representing knowledge about objects, especially shape and spatial relationships.
- Developing methods for reasoning about spatial relationships among objects
- Understanding the interaction between low-level information and high-level knowledge and expectations
- Rapid interpretation of stereo images

6. *Tactile Sensor Interpretation*

A two-dimensional tactile sensor provides an array of pressure measurements that corresponds to the gray-level intensity values found in a visual-image array. Thus, it is possible to use many of the image analysis techniques, such as edge analysis and delineation of regions, to analyze such information for object shape.

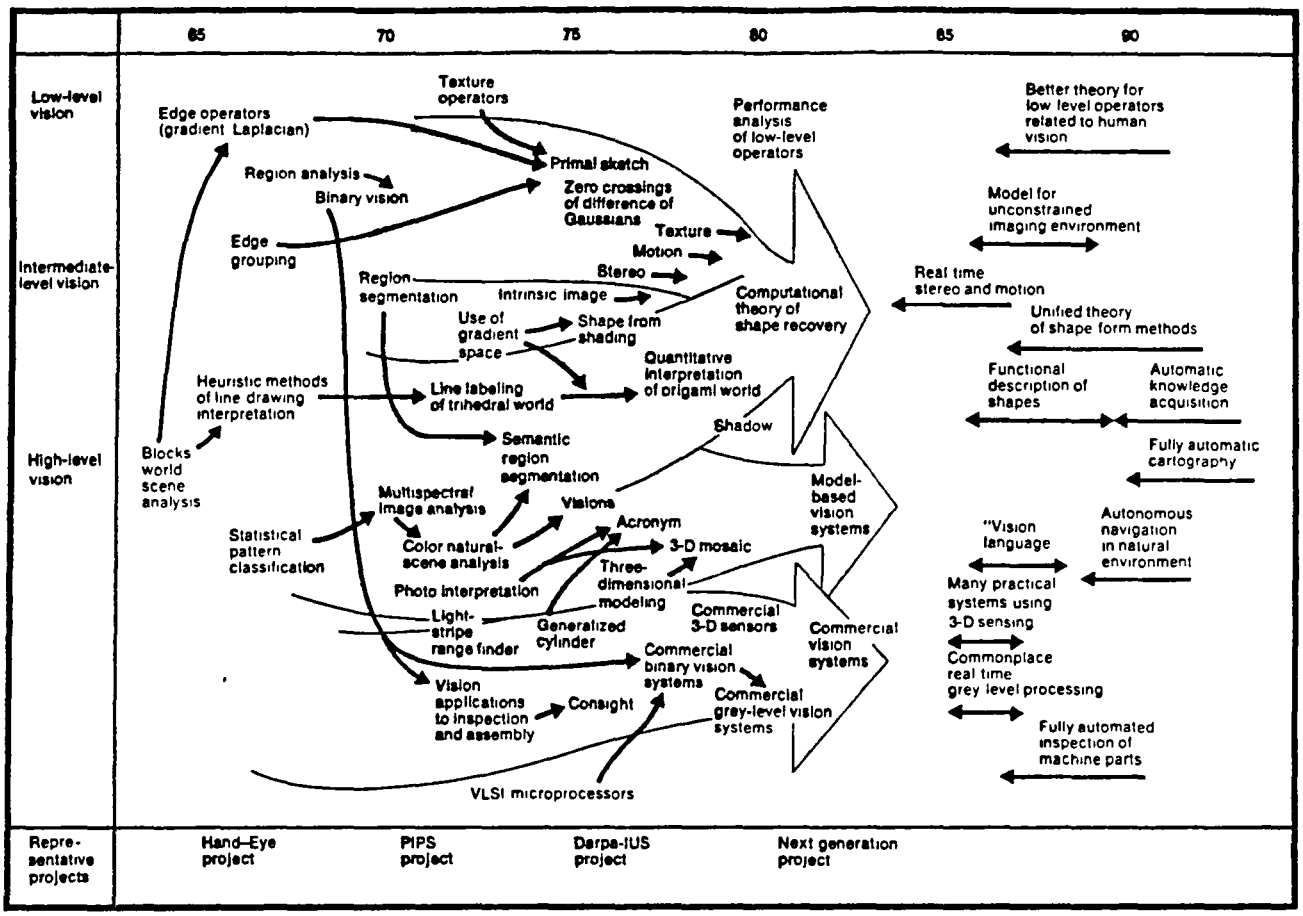
Since the tactile sensor is often exploring a surface or an object, the interpretation involves analysis of a *sequence* of pressure arrays. This corresponds to the sequence of image frames taken from a moving image sensor. Work in *optic flow* in the image domain may be pertinent here.

Research reports analyzing tactile sensor array data have recently appeared. The field will become more active when high-performance, low-cost tactile sensors become available to the general research community.

7. *Sensor Integration*

A person uses vision and touch in an integrated manner, utilizing cues from one sensor to aid in the interpretation of signals received from the other. To achieve high-performance robot manipulation, similar integration of multiple robot sensors will be essential, but there has been little research work done on

Figure 41: Progress in the Field of Computational Vision, (IEEE Spectrum, November 1983)



Computer vision efforts have advanced over the past 20 years along three fronts low-level vision, the extraction of basic features, such as edges from an image, intermediate-level vision, the deduction of the three-dimensional shape of objects from the images, and high-level vision, the recognition of objects and their relationships Some representative research projects include the Hand-Eye robotic vision project initiated at the Massachusetts Institute of Technology in Cambridge and at Stanford University in Palo Alto, Calif ; the pattern-information processing system (PIPS) project in Japan, one of the earliest focused research programs sponsored by the Ministry of International Trade and Industry; the U.S. Defense Advanced Research Projects Agency's image-understanding system (IUS) project, and the current Darpa next-generation project.

integrating vision and tactile sensors in robotics. DARPA's Strategic Computing Program, however, includes a strong component of sensor integration, particularly with respect to vision and range sensors.

C. Space Station Applications

The performance of servicing, construction, and manufacturing tasks by teleoperation in the same or less time than is possible by EVA presupposes a high degree of dexterity that can be achieved only with good visual and tactile sensing. The same sensory inputs will be needed in the autonomous mode to operate robot arms with equal dexterity. Machine vision will be easiest to apply in those space station activities in which the appearance of the work area is highly predictable. For example, it would probably be easy to guide the RMS grapple automatically to mate with a standard NASA docking probe, since the probe has a target designed to indicate any misalignment visually. A complex, cluttered environment such as an orbital construction site, however, would pose many still unresolved problems in computer image "understanding."

Force reflection from the manipulator hand to a teleoperator master control is one kind of tactile sensing that increases the operator's dexterity. The same information will be needed by a computer to make the gripper exert the forces and torques required for a task, as well as to ensure its proper compliance in response to external stimuli so that it will operate mechanisms and assemble components without jamming To handle small parts well, both teleoperation and robotic systems will have to sense not only that they are holding them, but also just where a part is in the fingers, how it is oriented, and (to identify it), what shape and size it is. To do this, we shall need small "fingertip" sensors that can measure pressure distributions with high spatial resolution over a planar region extending about one inch on either side. Simple proximity sensing (an elementary kind of range sensing) will also be useful—mainly to help avoid collisions between manipulators and other objects, but also to locate objects where visual or tactile sensing is impractical.

To make some of the sensor concepts more concrete, let us examine the sensor aspects of the scenario presented in Appendix A for parking and deploying the antenna boom. To carry out this task, an adaptive robot should be able to locate a known object by using vision, touch, or other senses. In addition, it should sense how it is grasping an object and control its gripper so as to maintain a firm grasp.

The sensor analysis system "knows" about the geometrical and physical characteristics of the antenna boom from a CAD data base. This data base would indicate the shape and size of the parts, how they are fastened and unfastened,

any special caution to observe while performing the operations, and how to overcome problems that arise. To locate the boom, the image understanding program would process the sensed image to find the edges and regions of the object viewed, then compare these with the edges and regions given in the CAD data base. The image understanding program is *model-driven*, i.e., the program can direct its attention to portions of the sensed scene in which it expects to find certain edges and regions. The output of the image-understanding program is used to guide the effector to the desired location. Thus, if the robot is to undo screws, the image understanding program must first locate them, and then specify to the effector control which movements are required to position a wrench or screwdriver on the screw.

The interaction between the effector control and the image understanding system is continuous, with the control subsystem applying the results of the interpretation system. In addition, because of lighting problems, the interpretation system may have to request that a different approach be made to the antenna boom, or that light be directed to a certain location.

For the adaptive robot to "understand" the nature of its grasp, the pattern of pressures and torques sensed in its tactile array must be interpreted. This problem is made more complex by the fact that a person often uses the sequence of sensed pressures to determine the texture, slip, or the nature of the object being grasped.

To rotate the assembly about the pins, it is important that the effector "follow" the natural motion of the boom. The CAD data base can be of some use, since it can indicate the boom's expected plane of motion. However, the dynamic sensory-pressure-pattern information will probably play a key role here. The robot control system will employ this dynamic information to "perceive" any increase in undesirable forces, whereupon it can correct the arm accordingly.

D. Research Funding

The Autonomous Land Vehicle (ALV) program of the DARPA's SCP has a strong vision system component. The functional objectives for this system are as follows

- FY86 -- Model and recognize simple terrain with crude objects.
- FY88 -- Recognize and match landmarks with maps, in simple terrain
- FY90 -- Same as FY88, but for complex terrain and using rich object descriptions.

- FY92 -- Perform reconnaissance in a dynamically changing environment.

The most significant technology resulting from this effort will be generic scene-understanding capability, plus the integration of sensors and automatic planning systems. However, the results in this area may not be entirely applicable to NASA's space environment problems because the ALV requires *passive* sensing methods, while the space station can (and should) use *active* sensing, special markings, and reflectors to simplify interpretation.

If the results of the vision research supported by DARPA's ALV program are made available, they could of course be utilized in NASA's sensor demonstrations. We have therefore concentrated on research and development that are not part of the DARPA program -- namely visual and tactile sensors that represent a transitional technology from teleoperation to more automated operation of robot arms. A basic goal of the research is to develop algorithms and techniques that will make possible the automatic understanding of complex objects under variable lighting conditions. Such a capability is essential for directing manipulator arms and effectors in the execution of a task. A CAD data base often plays a key role in achieving this level of understanding.

E Research and Development

The chronology of sensor research and development given below is based the teleoperation and robotics demonstrations described in Chapter VI. The dates signify when actual R&D results are expected to be needed:

- Incorporation of model-based visual analysis, requiring integration of the visual analysis system with a three-dimensional CAD data base (1987).

- Force and torque sensing (1988).

- Proximity sensing (e.g., using capacitance, dielectric phenomena, structured light, or—in pressurized areas—acoustic effects) (1988).

- Tactile sensing: development of sensors and analysis of sensor signals (1989).

- Rapid, high-resolution three-dimensional image analysis (1989).

- Integration of sensors of the same modality, as well as mixed modality such as visual, tactile, and range sensors (1992).

REFERENCES

1. W. B. Gevarter, "An Overview of Computer Vision," (Tech. Rep. NBSIR 82-2582), Washington, D.C., National Aeronautics and Space Administration, September 1982.

2. A. Rosenfeld, "Computer Vision Research for Industrial Applications," *Proc. Trends and Applications 1983: Automating Intelligent Behavior* (May 1983). IEEE Catalog No. 83CH1887-9.

3. L. D. Harmon, "Automated Tactile Sensing," *The International Journal of Robotics Research*, Summer 1982, *1*(2), pp. 3-32.

4. M. H. Raibert and J. E. Tanner, "Design and Implementation of a VLSI Tactile Sensing Computer," *The International Journal of Robotics Research*, Fall 1982, *1*(3), pp. .

5. W. D. Hillis, "A High-Resolution Image Touch Sensor," *The International Journal of Robotics Research*, Summer 1982, *1*(2), pp. 33-44.

6. S. Hackwood, G. Beni, L.A. Hornak, R. Wolfe, T.J. Nelson, "A Torque-Sensitive Tactile Array for Robotics," *The International Journal of Robotics Research*, Summer 1983, Vol. *2*(2).

7. J. Albus, "Proximity-Vision system for Protoflight," (Tech. Rep. NBSIR 78-1576), Washington, D.C., National Bureau of Standards, January 1979.

8. T. Okada, "Development of an Optical Distance Sensor for Robots," *The International Journal of Robotics Research*, Winter 1982, Vol. *1*(4).

VII. EXPERT SYSTEMS

VII EXPERT SYSTEMS

A. Introduction

The term *expert systems* was originally used to denote systems that utilize a significant amount of expert information about a particular domain to solve problems in that domain. Because of the important role of knowledge in such systems, they have also been called *knowledge-based* systems. However, the term has since been applied to so many diverse systems that its original meaning has been largely lost. There are essentially two uses of the term that need to be differentiated.

First, the term is often used to describe any system constructed with special kinds of "expert system" programming languages and tools. These include production systems, rule-based systems, frame-based systems, "blackboard" architectures, and the Prolog language. Unlike standard programming languages, these systems have very little control structure: functional components or modules are invoked primarily on the basis of properties of the current situation. Thus, each module usually consists of two components: the first describes those situations in which the module can be invoked; the second is the functional part to be executed whenever the module is indeed invoked.

The other important feature of such systems is that, since they are usually non-deterministic, a large number of modules may be "applicable" at any given moment. Thus, it is necessary to specify a criterion that determines how to select which of the applicable modules to execute next, and what to do after selection. Some systems select a single module to execute and use backtracking to allow other choices to be made, others apply modules in parallel, and yet others make irrevocable choices.

Development of "expert-system" languages is better regarded as an area of programming methodology or software engineering and, indeed, has made a significant contribution to these fields. However, it is very important to realize that such languages can be used for a variety of programming tasks apart from the construction of systems that emulate expert reasoning. Consequently, it is misleading to call any system developed in this manner "an expert system." Nevertheless, influenced by the considerable weight of accepted usage, we shall continue to call such languages (together with their supporting environments) "expert-system programming tools."

The second use of the term "expert system" is to describe any system that "reasons" about a problem in much the same way humans do. Some of the features distinguishing these systems from standard application programs are the following:

Knowledge—Each contains a data base of knowledge (usually in a specialized area) represented in a relatively natural form that allows some sort of reasoning to be carried out. The knowledge representations are usually symbolic, reflecting the qualitative nature of much human reasoning. Early expert systems used relatively simple rule-based or frame-based schemes for representing this knowledge. Recent systems have augmented these approaches by more sophisticated knowledge-representation formalisms.

Extensibility—The representation of knowledge is such that modifications of or additions to the knowledge base do not require extensive modification of the entire system. Thus the systems are extensible, degrade "gracefully" rather than catastrophically as elements are removed, and can evolve without extensive rewriting. This requires highly modular systems, in which the semantics of each module can be specified independently of other modules. Such evolutionary capability is essential for space station automation.

Flexibility—The systems are often highly reactive—that is, the choice of actions to be performed next by the system depends primarily on significant features of the current situation, rather than on the fixed and immutable control structure that characterizes more standard software systems. This is particularly important for space station applications, as the controlling systems must be flexible enough to respond rapidly to environmental changes.

Explanation—Many systems can retrace the reasoning sequence employed and explain what was done at each step and why. This explanatory capability enables the user to accept or reject the system's conclusions if he disagrees with its reasoning, and aids the expert in debugging the system. The usefulness of the explanatory system derives from the fact that the reasoning performed reflects the user's own reasoning processes.

Incomplete or Inexact Data—Many of these systems can carry out reasoning processes on incomplete, uncertain, or inaccurate data. For example, the effects of a given action may be incomplete, the conclusion of a diagnosis may be uncertain, or there may be errors in sensory information.

At present, expert systems do not acquire their expertise through experience, but are rather given the needed information by a "team" consisting of a programmer and an expert in the field. The programmer has come to be known

as a *knowledge engineer* because he must be familiar with the knowledge representation scheme used in the system. An expert system can be considered analogous to an *idiot savant* who can deal very effectively with a specialized field, but is incompetent to deal with topics outside its purview.

B. State of the Art

1. *Expert System Programming Tools*

There currently exist a number of expert system programming tools. These include OPS [1], S1 (Teknowledge), ART (Inference Corp.), ROSIE (RAND Corp.), and KEE (Intellicorp). All the available "off-the-shelf" systems are little more than programming languages as discussed above — namely, they provide a programming language (usually with a very rich support environment) that is suited to constructing expert systems. However, just as there is a lot of work involved in writing an Ada program to perform some complex analysis, so is there at least as much work in constructing an expert system with an expert system programming tool.

While the available expert-system programming tools are well suited to developing expert systems that require relatively simple knowledge representations, it is not at all clear that they are useful in handling the more powerful and expressive knowledge formalisms needed for more complex problem domains. Indeed, they can actually hinder development in these areas. It is usually better to build the more complex formalisms upon a more basic programming language.

LISP and Prolog are ideal for this purpose. Both languages serve different needs and it is likely that any reasonably sophisticated system will have to use both. In particular, Prolog is useful for representing knowledge that is naturally expressed as a set of facts and a set of rules, with the rules serving to define how new facts are to be deduced from what is already known. LISP is well suited to procedural programming and to implementing more complex knowledge representations in the form of list or network structures.

2. *Expert Reasoning Systems*

The general categories of tasks that expert systems have been applied to can be broken down as follows:

- Interpretation and Diagnosis—This category of expert systems includes all those that can accept data from the user about a particular case and, when sufficient information has been received, return a diagnosis or interpretation of that case. Examples include mass-spectrometer data interpretation

(DENDRAL [2]) and medical expert systems (EXPERT [3], MYCIN [4]). Systems for fault diagnosis and isolation also fall into this class. Some current systems are listed in Table 3.

- Design Systems—These are expert systems that may be given particular information and constraints and are required to produce an output that satisfies the given design criteria. An example is XCON [5], an expert system that designs computer configurations.

- Prediction and Induction Systems—These systems accept data and look for patterns or other forms of order. When such patterns are found, they can be combined with information about a particular case to predict the most likely outcome. An example of an inductive system is INDUCE, which infers the relationship between symptoms and disease in soybeans.

- Monitoring and Control Systems—These systems receive specific online data from sensors regarding the object being monitored and/or controlled. These data are rapidly interpreted by the expert system and the appropriate responses generated. In a monitoring expert system, specified alarms are triggered whenever particular critical situations are detected. REACTOR, a nuclear-reactor-monitoring system, and VM, a patient-monitoring system for intensive care wards, are examples of this type of expert system. YES/MVS, an IBM system, is an example of a real-time expert system used to control an operating system. Often the generation of an appropriate response will require simulation of the expected effects of possible actions on the controlled system.

Any premature enthusiasm over the apparent success of these systems needs to be tempered by the following observations. First, very few such systems have been developed beyond the experimental testing stage. Although such testing is essential in establishing the soundness of the basic design, there can still arise serious technical problems in getting the system to work in a real environment.

Second, most of the expert systems developed to date cannot easily be generalized to handle problem domains other than the ones they were specifically designed for. In other words, each of them is an application program that was designed and constructed for one particular application.

Third, the kinds of knowledge that existing systems can represent are relatively simple. This does not mean that they are not useful, but it does mean

that the application of expert systems to more complex domains will require a significant amount of research in knowledge representation.

C. Present Limitations of Expert Systems

There are limitations in present-day expert systems because of the following technical problems:

- Knowledge representation. It is difficult to develop representations for specific domains that are computationally tractable and still capture the important characteristics of the domains. In particular, formalisms to represent time, space, actions, processes, mechanisms, and other complex objects need to be developed (e.g., [13]). While considerable attention has been focused on static problem domains, and on capturing an expert's knowledge in the form of heuristic rules of thumb, relatively little attention has been applied to dynamic domains where much expert knowledge is procedural—that is, where expert knowledge involves reasoning about sequences of tests and actions. Another problem is how to represent "commonsense" knowledge—the type of knowledge a person uses in dealing with the world. Unless this type of knowledge is incorporated in expert systems, they will remain "fragile," i.e., unable to function except when dealing exclusively with their narrow specialties.

- Reasoning. The reasoning system must be able to reach conclusions on the basis of information about the current situation and the knowledge contained in the knowledge base. Much work is required to develop techniques for qualitative and quantitative reasoning. Techniques are required for reasoning on the basis of uncertain information and weakly supported implications, for updating the knowledge base over time, and for maintaining its consistency.

- Knowledge acquisition. There is a major problem in obtaining, representing, and debugging expert knowledge about a particular domain. Even for the best-understood problems, typically about five man-years of effort are required to develop a large system that begins to be robust. Methods are now being developed for dealing with these problems that should reduce the time it takes to build new systems.

- Verification. Since the system may be inconsistent in its knowledge or rules, it is important that manual and eventually

Table 3: Existing Expert Maintenance Systems

- EL, an MIT program, simulates the operation of an electrical circuit and deduces the possible cause of a failure [6].

- IN-ATE, Navy Center for Applied Research in Artificial Intelligence, is an expert system for guiding a novice technician in troubleshooting electronic equipment [7].

- MDIS (Maintenance-and-Diagnostics Information System), Boeing Aerospace, is an expert system for maintenance and diagnosis [8].

- IMA (Intelligent Maintenance Aid), General Dynamics, is a prototype expert system for diagnosis of the microwave stimulus interface (MSI) of the F-16 Avionics Intermediate Shop [9].

- DART, an ongoing, joint, IBM-Stanford University project, uses a causal model of a computer for fault diagnosis.

- DELTA, or CATS-1, is an expert system developed at GE for trouble-shooting diesel-electric locomotives [10].

- ACE (Automated Cable Expertise), Bell Laboratories, identifies trouble spots on the basis of data from trouble reports, and suggests the repairs to be made [11].

- LES (Lockheed Expert System), developed by the Lockheed Palo Alto Research Laboratory, is a general-purpose expert system that has been applied to diagnosing faults in a complex switching network.

- PES (Procedural Expert Systems), developed at the AI Center of SRI, is a system for space station maintenance that explicitly represents procedural knowledge while retaining the benefits of traditional expert systems [12].

automatic verification techniques be developed. One approach is to provide users with knowledge-based debugging tools that, for example, check for inconsistencies and gaps in the knowledge base and help the experts and knowledge engineers to communicate with one another [14]. However, the problem of formally verifying a knowledge base constructed with an expressive representation scheme is, in general, intractable. Considerable research will therefore be necessary to extend the limits of current verification techniques.

- Explanation capabilities. The explanations produced by current expert systems are usually indications of the solution path traversed before attaining the present status, rather than being causal explanations of the type people usually provide. However, what the user often desires is a causal explanation based on physical reasoning. This type of explanation must be based on a very rich description of the problem domain, for which a representation of the model or mechanism underlying the reasoning is normally essential. Furthermore, there is difficulty in making explicit the information that may be implicit in the representation, then forming it into an explanation that is acceptable to the user.

- Use of metaknowledge. The system should have knowledge about the knowledge it contains, and be able to use such knowledge in its reasoning strategy. Such *metaknowledge* becomes crucial when large knowledge bases are to be used, since otherwise too much time is expended on unproductive searches, inappropriate actions, and needless data requests.

- Learning capability. Currently the designer of the system, not the system itself, learns by experience as the system is used. Thus, the designer, not the system, modifies the knowledge base. It is a nontrivial task to determine which rules need modification when the expert system is not performing up to an expert's standards. The designer must consult with the human experts to determine how the rules have to be modified or augmented. The system itself has no way of determining that the user is dissatisfied, nor of automatically correcting the source of the difficulty. It is important that techniques be developed for acquiring knowledge automatically as the system performs its tasks.

D. Space Station Applications

Some application areas of expert systems in the space station are as follows:

- Maintenance and repair. Expert systems are important in subsystem and satellite servicing, for carrying out routine tests, noting possible deviance, and flagging abnormal transient operation before a hard failure occurs. In addition, expert systems will be needed to isolate and diagnose faults, and to indicate methods of handling malfunctions.

- Expert process controller. In manufacturing, expert systems are required for quality assurance (interpreting process deficiencies), process control (suggesting processing corrections to attain better results), and equipment maintenance (isolating equipment faults and initiating corrective action).

- Subsystem monitoring and control. Expert systems can be applied to subsystems, such as the power subsystem, to monitor and control complex operations and make difficult decisions. Maintenance of life support systems, operation and servicing of experiments, onboard mission control, and automation of traffic control could also be handled by expert systems.

- Intelligent autonomous robots. An expert system could guide the scheduling of the construction and assembly of large space structures, the servicing of satellites, deployment of payloads, OMV/OTV operations, and the transfer of cryogenic fluids. Eventually, as effector and sensor capabilities are developed, these processes could be automated and handled in their entirety by autonomous robots.

- Astronaut's associate. An expert system could act as an astronaut advisor to aid in the use of a complex program or a complicated item of equipment. The advisor could suggest parameter values, the meaning of certain system responses, and sequences of control actions.

In many of these areas, there will be some subclasses of problem that can be solved by constructing simple expert systems that use relatively elementary knowledge representation schemes. Commercially available expert-system programming tools may be adequate for creating such systems, while the deeper problems of some applications would at least be indicated as targets for future resolution. Furthermore, there are some applications, such as simple monitoring

and control, for which current tools could be used advantageously, even though the resulting systems might not reflect any expert reasoning at all or provide any useful explanatory capabilities.

However, the more complex space station tasks require expert systems capable of sophisticated reasoning about actions, events, and processes. Typical of the kind of knowledge used in these applications are the malfunction handling procedures for the space shuttle (STS). Sample portions of these procedures are given in Figure 42. As can be seen, the procedures are extremely complex, and involve performing sequences of actions and tests that change the state of the space station and its environment. Furthermore, the nature of this knowledge is procedural — that is, it is represented in the form of complex procedures for achieving given goals rather than as a set of "rules" about shuttle operations.

A schematic view of an expert system suited to these applications is shown in Figure 43. The central role of reasoning, involving a knowledge base and a reasoner and planner, is indicated. The reasoning portion receives information about the world both from the system interfaces (communication) and from the sensors (control/sensing), and integrates it with the information in the knowledge base using the consistency maintainer. The role of the consistency maintainer is to ensure that changes to knowledge base entries do not cause inconsistencies. The output of the reasoning portion is used to communicate with other systems and to generate commands to the effectors and sensors

The development of such systems could only be brought about by pursuing a well-focused research plan investigating the critical issues involved in knowledge representation and reasoning. Unless this is done, it is difficult to see any possibility of automating space station functions; furthermore, expert systems will only find useful application in a few relatively simple tasks. A research plan to develop expert systems for space station needs is described later in this chapter.

E Research Issues

Research in expert systems should be focused on basic research and specialized development for the needs of the space station—i.e , for use in the operation and control of space station subsystems and manufacturing processes. This section delineates specific areas of research relating to expert systems for these purposes Each topic is described in general, the purpose or end product is indicated, and specific research tasks are described.

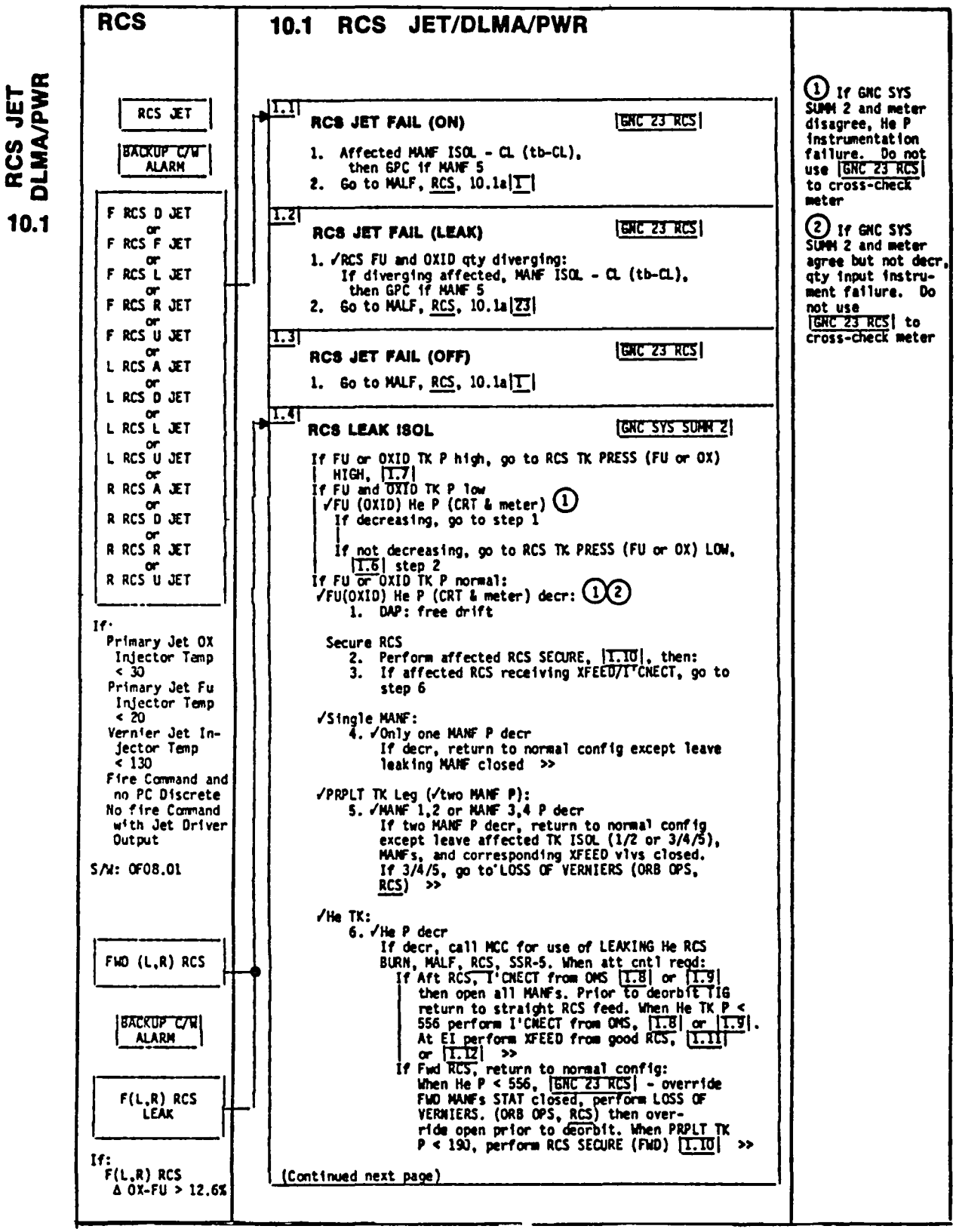


Figure 42: A Typical Malfunction Handling Procedure for the STS

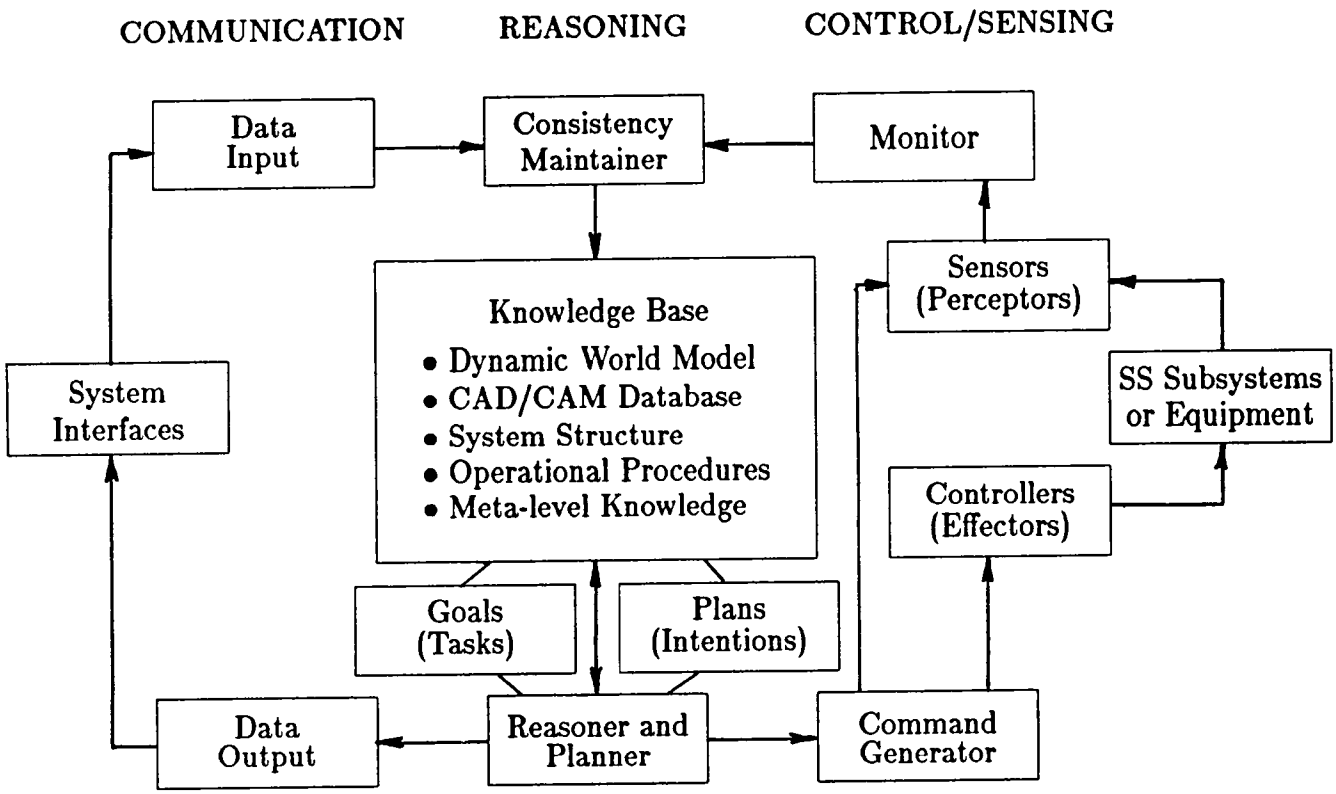


Figure 43: Expert System for the Space Station

1. *Knowledge Representation*

General description. The representation of knowledge is a central issue in expert systems, as it constitutes the basis for reasoning and explanation. The representation chosen depends on the nature of the application, but it is important that it be flexible, verifiable and extensible. These are some aspects of knowledge representation that require substantial research:

- The representation of actions and events, causality, the effects of sequences of actions, and concurrency.
- The representation of the structure and function of physical mechanisms, and how this can be used to reason about behavior.
- The use of multiple representations (e.g., physical and electrical),

and multiple levels of abstraction.

- The representation of procedural knowledge, where knowledge is expressed in terms of [possibly complex] procedures or sequences of tests and actions for achieving various goals (e.g., as occurs in operational procedures).

- The representation of "metaknowledge" (knowledge about the knowledge in the data base), and how it can be used to improve the performance of the system.

End product. The end product would be both general-purpose and application-oriented knowledge representations suitable for various space station applications

Specific research tasks. Research in knowledge representation should be carried out in the context of both specific space station applications and general representational issues. One critical feature of most expert-system applications on the space station is that the problem domain is dynamic, i.e., actions and events take place that change the state of the world. Little work has been done on expert systems that reason about dynamic domains and can operate effectively in real time; thus, it is important that this area of research be expanded as much as possible. Schemes for explicitly representing and using expert knowledge of a procedural kind must be developed The representation needs to be rich enough to describe a wide class of actions and plans simply and naturally, and in a way that is flexible, extensible, and verifiable. Because multiple subsystems on the space station will be running in parallel, it is important that the representations allow reasoning about concurrency and cooperation. To handle the more difficult problems that arise in space station applications, it is also important to be able to represent and reason about structures, mechanisms and processes. Schemes must also be developed for representing and reasoning about geometrical properties of objects and continuous (rather than discrete) time.

2. *Reasoning, Inference, and Uncertainty*

General description. It is important to be able to reason on the basis of incomplete or uncertain evidence, and to reason about time, actions, mechanisms, and physical processes. Research in this area would result in techniques that are suitable for such reasoning and are focused on specific applications. Such techniques would be concerned with updating the data base over time, handling data errors or inconsistencies in the data base, reasoning about the "beliefs" of other subsystems, and reasoning about interactions among subsystems.

End product This research would provide reasoning systems capable of

handling uncertain data and dynamic domains.

Specific research tasks. There is a need to investigate fast theorem provers for drawing inferences in real time and handling a rich class of logical expressions. Questions such as maintaining the consistency of the knowledge base and the coordination of distributed knowledge must also be addressed. There is a need to investigate methods of reasoning about interaction among processes, and to be able to reason about multiple and cooperating subsystems. The role of Bayesian techniques, evidential reasoning, fuzzy reasoning, and nonprobabilistic approaches to managing incomplete or uncertain data must be studied.

3. *Knowledge Acquisition and Verification*

General description. In developing an expert system, the initial acquisition of knowledge is followed by its subsequent refinement. Knowledge verification is essential at both stages.

End product. The ultimate goal is to allow an expert to encode his own knowledge directly without any need for a knowledge engineer, to refine that knowledge, and to verify the correctness of the knowledge base.

Specific research tasks. The following research tasks would be appropriate:

- *Initial knowledge acquisition.* Since much of the role of the knowledge engineer is to aid in the structuring and formalization of the problem domain, it is not clear that this can be done automatically. An appropriate research task would be to develop a menu of models that would be useful for a particular class of applications. The expert, in a dialogue with the system, would select the most appropriate knowledge formalism. Natural-language interaction can be important here.

- *Knowledge verification.* The builders of a knowledge-based expert system must ensure that the system will give its users accurate advice or correct solutions to their problems. The process of verifying that a system is indeed accurate and reliable involves testing and refining the system's knowledge to discover and correct various errors that can arise in the process of transferring expertise from a human expert to a computer system.

- *Knowledge refinement.* An important tool for refining a knowledge base is a facility for displaying the existing base in some graphic or structured manner. Tools need to be developed for interacting with an expert to indicate gaps in the data base for which knowledge or procedures should be supplied.

4 *Explanation*

General description. There are various kinds of explanations that might be provided in an expert system:

- What is the immediate goal?
- What is the justification for an action?
- How will a particular action contribute to the goal?
- Why is that a good choice of methods to accomplish the goal?
- How does the mechanism work?

End product. The end product of this research would be techniques for supporting various types of explanation

Specific research tasks. The various space station applications need to be examined to determine the kind of explanation that is most useful There should be schemes for providing richer information about the reasoning being used and for giving causal descriptions of processes. It may be necessary to develop special representations for domain-specific applications. The research should also explore the problem of converting an internal representation to a form that is more manageable for the user, such as natural language and sophisticated graphic displays.

5. *Distributed Architectures*

General description. To maximize evolutionary capability and enhance real-time performance, it is important to provide for distributed processing by multiple expert systems. These systems might control and operate various subsystems (such as power, life support and experiments), or they might be distributed among a central higher-level reasoning system and its individual sensors and controllers.

End product. A distributed expert system capable of fast real-time response.

Specific research tasks. In developing a distributed expert system, it is necessary to be able to reason about the knowledge (or, more generally, the "beliefs") of other component expert systems, of their goals and plans. It is also necessary to communicate with other systems and be able to synchronize activities, so as to avoid harmful interactions and provide for cooperation. In more complex systems, it may be necessary to reason about possibly inconsistent information from different systems (such as when two sensing systems give

different results), and to construct methods for resolving such inconsistencies (such as making comparisons with other data). There has been very little research in these areas to date.

6. *Learning*

General description. There is no reason to expect that the addition of incremental amounts of knowledge will lead to global understanding. To be able to *understand*, the system must be able to generate higher-level concepts by comparing and generalizing problematic situations or groups of knowledge elements. No such capability has been provided in existing systems. This ability to make comparisons and establish similarities is a crucial part of generalization and learning.

End product. An expert system that learns from experience, just as a human expert does in the process of broadening his knowledge.

Specific research tasks. Since this is a very difficult research area at the present time, relatively simple tasks should be attempted at first. An initial project would be to develop inference systems capable of classifying patterns of behavior, such as must be done in analyzing astronaut's medical data for an in-flight health care system.

7. *Space-Qualified Symbolic Processors*

General description. Space-qualified symbolic processors are required for dealing with the types of symbolic manipulation often found in AI applications

End product. Small, space-qualified VLSI symbolic processors for use in space station expert systems and other AI applications

Specific research tasks Research should be conducted on constructing VSLI symbolic processors for use in space, developing key hardware components for symbolic computing—including retrieval based on unification and partial pattern matching—and investigating highly parallel architectures.

8. *CAD Data Base Requirements*

General description. An expert system that reasons about objects and actions in the world must have a model of the objects it deals with and the actions it can perform. The CAD data base would provide this type of information It would do so by indicating characteristics of space station systems and equipment, and operational and maintenance procedures.

End product. The definition of requirements for a data base suitable for use by expert systems, and possibly a uniform data base language that is sufficiently expressive for the variety of systems and equipment on the space station.

Specific research tasks The first task should be to identify the nature of the data base information required. Such information should include CAD/CAM specifications, structural and functional descriptions of the space station systems, and as much design data as possible. It should also include details on operating and maintenance procedures, with annotations describing the purpose of every routine and of each step within the routines. To the extent possible, this information should be couched in a uniform formal language, such as predicate calculus or some more natural variant. The data base system itself should allow very fast access and should contain mechanisms for addition and deletion.

F. Research Funding

1. *Research and Development by Others*

The DARPA Strategic Computing Plan will have a major influence on expert system development. DARPA believes that the most time-consuming portion of the process of constructing an expert system is the expert's articulation of his knowledge, and the formulation of this knowledge in a suitable representation language. The SCP therefore places particular emphasis on knowledge acquisition and representation. In its own words:

The SCP expert system technology effort will generate and extend AI techniques by improving software support tools and by using specialized symbolic computational hardware. Work in representation will build toward a capability for a 30,000-rule knowledge base. Inference techniques will be extended to handle these knowledge bases even when they contain uncertain knowledge and must operate on erroneous and incomplete data. The knowledge acquisition work will focus on developing facilities for automated input of domain knowledge directly from experts, text, and data. It is estimated that hybrid expert system architectural configurations will be required that can accommodate 30,000 rules and perform at a capacity of 12,000 rules per second. This architecture will be realized through VLSI devices incorporating massive parallelism, active semantic memories, and specialized inference mechanisms The resulting technology will be substantially generic in nature so that it will significantly advance expert system capabilities and support a wide range of applications for both government and industry.

The DARPA development plan is as follows:

- 1986-1987—3000 rules with 1000 rule inferences per second (RIPS); 1/3 to 1/2 real time; moderately difficult
- 1989-1991—10,000 rules with 4000 RIPS; real time; complex context

- 1992-1993—30,000 rules with 12,000 RIPS; 5 times real time; highly complex context

Clearly, any NASA research program in expert systems will have to take into account the research being carried out in the DARPA program. In particular, the work on computer architectures to support expert systems is very important. However, the DARPA emphasis on "rule-based" approaches is probably misguided; it is rather the concepts of flexible, extensible, and reactive systems that are crucial. Rule-based schemes are not the only (and often not the best) way of achieving these goals. Moreover, it is far from clear that the "rules" used in these schemes are the best representational form for more complex kinds of knowledge and reasoning.

The DARPA Pilot's Associate portion of SCP will also contribute to the development of expert systems. The Pilot's Associate is an intelligent system that assists the pilot in the air as well as on the ground. It does not replace, but rather complements the pilot by relieving him of lower-level chores and performing special functions so that he may concentrate his intellectual resources on tactical and strategic objectives. The system would draw upon an extensive knowledge base of information concerning such items as aircraft systems, tactics and strategy, and navigation. The three aspects of this project that could be relevant to the space station are the interface with the pilot, the organization of multiple interacting expert systems and the supporting knowledge bases, and the high-speed processors. DARPA is currently evaluating proposals on this project, work on which has not yet begun. However, its main thrust will be in the area of situational assessment and navigation, rather than subsystem operations.

Other DoD efforts include a flight control maintenance-and-diagnosis system (Wright-Patterson Air Force Base) and fault isolation of army aviation systems (U S. Army Research and Technology Laboratories).

There is a great deal of industrial activity in the development of expert systems for practical use, although few of them require more than very simple knowledge representation schemes.

2. *Existing NASA Research and Development*

There has been considerable interest in expert systems within the NASA centers. Some of the expert systems being developed are:

- The LOX Expert System [15], a knowledge-based system being applied to a semi-real-time application monitoring the loading of cryogenic fuel for the space shuttle The KNOBS constraint- and frame-oriented expert system is being used.

- FAITH (Forming and Intelligently Testing Hypotheses) is a JPL system for automating the Voyager downlink process.

- An expert system for supporting space shuttle flight control has been proposed [16].

- PES, a procedural expert system for space station maintenance, is being supported by NASA Ames Research Center [12, 17].

In addition, several NASA centers are in the process of developing expert systems for various applications, including fault diagnosis of the life support system and distribution of electric power. Most of these applications utilize the results of expert-systems research conducted at universities during the past decade, and perform relatively simple tasks that can be achieved with quite weak knowledge representations.

3. *Recommended NASA Research and Development*

As indicated above, there is a considerable amount of research and development being carried out in expert systems, with major funding support available in the DARPA program. In delineating the work done in expert systems, we have therefore focused our attention on basic research and specialized development for the needs of the space station. Our examination of space station applications has made it evident to us that the highest return on research investment is to be found in automation of the operation, maintenance, and control of space station subsystems and manufacturing processes. The crucial characteristic of these applications is that the domain is dynamic — i.e., it involves reasoning about the effects of sequences of actions and tests that can change the state of the world over time. Moreover, because various subsystems will be operating simultaneously, it is important that the representation be sufficiently rich to enable reasoning about concurrency and subsystem interaction, and that efficient procedures for automatic scheduling and synchronization be developed.

Very little research is being done in this area. Consequently, without NASA support it is unlikely that the technology necessary for automating space station operations could be developed before the end of this century. Furthermore, most of the research issues that arise in representing and reasoning about these applications are also of critical importance in developing intelligent robots. Thus, through concentration on generic formalisms, schemes for representation and reasoning can be devised that would be eminently suitable for both areas of application. In addition, such generic research would produce major benefits for terrestrial applications, both military and civilian.

As discussed elsewhere in this report, it is very important that the space station systems possess the potential for evolutionary growth. To achieve this, as well as to satisfy the real-time requirements of operating and control systems, a distributed architecture containing multiple expert systems must be employed. Since no existing expert system can meet these needs, considerable research will be needed to solve the related technical and theoretical problems.

Researchers developing symbolic processors can benefit from DARPA's research effort, adapting any useful results to the needs of the space station. Reasoning about uncertainty, a common feature of many current expert systems, plays a relatively small part in most space station applications. Work in this area, therefore, can be left principally to DARPA and industry. On the other hand, research on qualitative reasoning and on knowledge formalisms that are capable of representing structure, function, and mechanism will be vital to the long-term needs of space station automation. But this more essential research can be deferred until later in the program, thus enabling it to benefit also from similar work being conducted by DARPA.

G. Demonstrations

The following demonstrations would verify that the necessary capabilities are available; the proposed schedule is given in Figure 43. These demonstrations can start with ground simulations; they would next proceed to actual implementation—first on the shuttle, then on the space station itself.

Near-term (1985-1992)

- Information retrieval from a data base that describes the structure and functionality of major systems in formal or semiformal language.

- Information retrieval from a data base describing maintenance and operating procedures of major systems in formal or semiformal language, including information as to purpose of the procedures and their component steps.

- A system capable of fault isolation of a single subsystem, using standard maintenance procedures.

- Same as the preceding, but using distributed expert-system architecture with the aim of improving real-time performance and evolutionary potential

- A system capable of fault isolation of multiple interacting

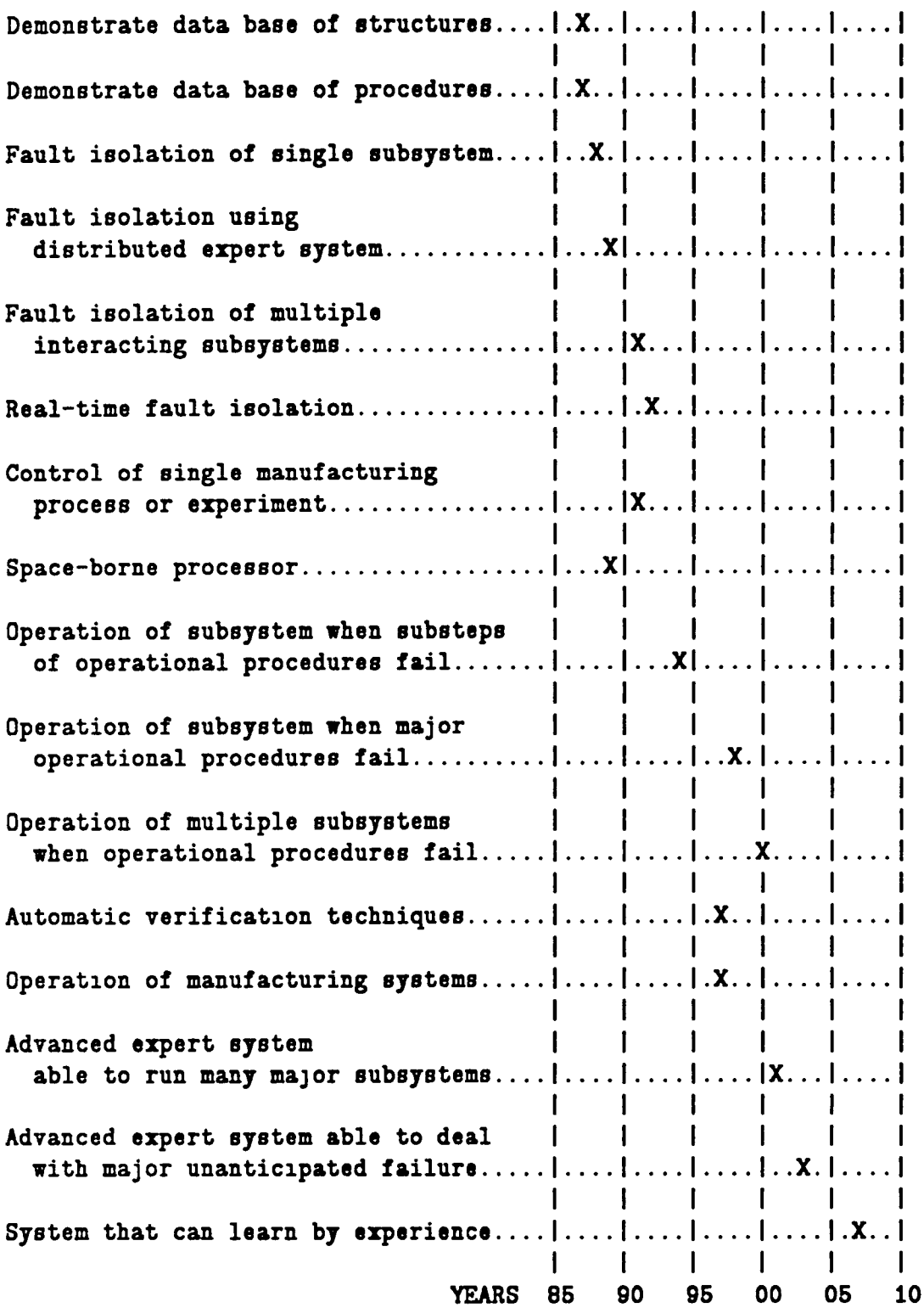


Figure 44: Schedule of Expert Systems Demonstrations

subsystems, using standard maintenance procedures.

- Same as the preceding, but operating under real-time constraints and allowing for data errors.

- A system for control of a single manufacturing process or a single experiment.

- A spaceborne processor particularly suited to mechanization of expert systems.

Medium-term (1993-2000)

- An expert system capable of solving problems in an isolated subsystem when some substeps of a standard maintenance procedure are inapplicable.

- An expert system capable of fault diagnosis and recovery in an isolated subsystem when a major portion of some standard maintenance procedure is inapplicable.

- Same as the preceding, but involving multiple interacting subsystems.

- Automatic-verification techniques for guaranteeing that an expert system is "safe," i.e., cannot harm the subsystems that it controls.

- An expert system for use in manufacturing, capable of limited quality control, production control, and maintenance and fault diagnosis of a manufacturing process.

Long-term (2001-2010)

- An advanced expert system that can run many major subsystems, maintain and control experiments and manufacturing processes, schedule tasks, and interact with intelligent robots.

- An advanced expert system that can cope with an unanticipated major system failure (like the one that occurred during Apollo 13).

- An expert system that can improve its own maintenance skills—i.e., "learn" from experience

H. Research and Development

To implement the sequence of demonstrations outlined above, the following research topics would require NASA funding:

Near-term (1985-1992)

- Techniques for representing and reasoning about procedural knowledge, particularly in conformance with the NASA style of describing operational procedures.
- Representation of actions and events; causality; reasoning about the effects of sequences of actions.
- Techniques for ensuring consistency of the knowledge base over time (truth maintenance, frame problem).
- Techniques for distributed systems; communication protocols; reasoning about the knowledge bases of other systems (mutual belief); communicating to exchange information.
- Techniques and representations suitable for reasoning about the reasoning process.
- Techniques for reasoning about concurrency and about interactions among subsystems; synchronization of processes.
- Techniques for reasoning about inconsistent information and data errors.
- Interactive techniques for verifying the correctness of expert systems.
- Space qualification of symbolic processors and development of special parallel architectures for expert systems.
- Fast theorem provers and rapid data base access and updating for supporting real-time reasoning.
- Delineation of the CAD data base and representation of maintenance procedures for expert maintenance systems; development of formal languages to represent this type of information.

Medium-term (1993-2000)

- Knowledge representation and reasoning techniques for dealing with the structure and function of physical mechanisms.
- Reasoning about deadlock, cooperation, and communication among multiple expert systems.
- Integration of qualitative and quantitative reasoning.
- Reasoning about geometric properties of objects and continuous time (as opposed to discrete time).

Long-term (2001-2010)

- Fully automatic techniques for verifying the correctness of expert systems.
- Representation of commonsense knowledge.
- Learning from past examples and reasoning by analogy.

VII. EXPERT SYSTEMS

REFERENCES

1. C. Forgy and J. McDermott, "OPS, a Domain-Independent Production System Language," *Proceedings of the 5 th International Joint Conference on Artificial Intelligence*, pp. 933-939, Cambridge, Massachusetts (August 1977).

2. B.G. Buchanan and E.A. Feigenbaum, "DENDRAL and Meta-DENDRAL: Their Applciations Dimensions," *Artificial Intelligence*, 1978, *11*, pp. 5-24.

3. S.M. Weiss and C.A. Kulikowski, "EXPERT: A system for developing consultation models," *IJCAI*, pp. 942-947 (1979).

4. B.G. Buchanan and E.H. Shortliffe. *Addison-Wesley Series in Artificial Intelligence. Rule-Based Expert Systems.* Reading, Massachusetts, Addison-Wesley, 1984.

5. J. McDermott and C. Forgy, "R1: An expert in the computer systems domain," *AAAI 1*, pp. 269-271 (1980).

6. R. Davis, "Diagnosis Via Causal Reasoning: Paths of Interaction and the Locality Principle," *Proceedings of the 3 rd National Conference on Artificial Intelligence*, pp. 88-94, Washington, D.C. (August 1983).

7. R.R. Cantone, F.J. Pipitone, W.B. Lander, and M.P. Marrone, "Model-Based Probabilistic Reasoning for Electronics Troubleshooting," *Proceedings of the 8 th International Joint Conference on Artificial Intelligence*, pp. 207-211, Karlsruhe, West Germany (August 1983).

8. D R. Antonelli, "The Application of Artificial Intelligence to a Maintenance and Diagnostic Information System (MDIS)," *Joint Services Workshop on Artificial Intelligence in Maintenance, Volume 1: Proceedings*, Boulder, Colorado (1983). in press.

9. J.H. Hinchman and M.C. Morgan, "Application of Artificial Intelligence to Equipment Maintenance," *Joint Services Workshop on Artificial Intelligence in Maintenance, Volume 1: Proceedings*, Boulder, Colorado (1983). in press.

10. H.E. Johnson, and P.P. Bonissone, "Expert System for Diesel Electric Locomotive Repair," *The Journal of Forth Application and Research*, September 1983, *1*(1), pp. 7-16.

11. G.T. Vesonder, S.J. Stolfo, J.E. Zielinski, F.D. Miller, and D.H. Copp, "ACE: An Expert System for Telephone Cable Maintenance," *Proceedings of the 8 th International Joint Conference on Artificial Intelligence*, pp. 116-121, Karlsruhe, West Germany (August 1983).

12. M. Georgeff, "Procedural Expert Systems," (AIC Technical Note 314), Menlo Park, California, Artificial Intelligence Center, SRI International, December 1983.

13. *Artificial Intelligence*, 1984, Vol. *24*(1-3). Special volume on qualitative reasoning about physical systems.

14. B.G. Buchanan and E.H. Shortliffe, "Completeness and Consistency in a Rule-Based System," In *Addison-Wesley Series in Artificial IntelligenceRule-Based Expert Systems.* Reading, Massachusetts, Addison-Wesley, 1984.

15. E.A. Scarl, J. Jamieson, and C. Delaune, "Knowledge-Based Fault Monitoring and Diagnosis in Space Shuttle Propellant Loading," (Interim Report), Cambridge, Mass, MITRE, 1984.

16 J. Helly, W. Bates, and M. Cutler, "A Representational Basis for the Development of a Distributed Expert System for Space Shuttle Flight Control," (Interim Report), Los Angeles, UCLA, 1984. NASA JSC.

17. M P. Georgeff, "An Expert System for Representing Procedural Knowledge," *Joint Services Workshop on Artificial Intelligence in Maintenance, Volume 1: Proceedings*, Boulder, Colorado (1983). in press.

VIII PLANNING

A. Introduction

Much of human problem-solving is devoted to planning and reasoning about actions designed to attain a goal. In everyday life we carry out such planning without much conscious thought — for example, when driving from one location to another or planning a shopping expedition. Because planning is such a natural part of human intelligence, we tend to overlook the complex problems that must be solved, particularly in a dynamic world filled with other "agents" who can affect "the best-laid plans of mice and men." The difficulties of planning increase with the complexity of the task. Planning all the activities that are necessary to construct and launch the space station, for example, involves finding and coordinating thousands of separate subtasks, so that the resultant plan will have no inconsistencies and will meet its specifications. Producing such plans manually requires thousands of man-hours, with no guarantee of correctness. Automating the process of plan generation would go a long way toward removing such bottlenecks.

We encounter two types of planning problems in the space station: (1) scheduling predefined activities, and (2) the planning of tasks. In the scheduling problem, one begins with a set of known activities, conditions to satisfy, and goals to achieve. In essence, the activities must be scheduled so as to make the best use of resources and satisfy the goals as well as possible. An example of this is the planning done by astronauts to accomplish the most important tasks, meanwhile coordinating these with other activities of the space station.

In task planning, the goals to be achieved are given, but the actions required to achieve them are not. A task planner must select appropriate actions and incorporate them into a plan for achieving the given goals. Often an initial set of "high-level" actions is given (a rough plan), to which detailed actions are then added that have been selected by the planner. During execution of the plan, replanning may be necessary if unexpected situations are encountered. Task planning is required, for example, in deciding how to move from one location to another, in determining what actions are necessary to repair or replace a failed component, or in enabling autonomous robots to carry out operations in conjunction with crew members or with other robots.

Task-planning systems can logically be divided into the following classes:

- Planning by a Single Controller. Given certain specific goals, the planner synthesizes a plan for control of a single agent, or multiple agents under a single controller. This plan is created by determining which actions achieve the goals, and in what sequence the actions should be performed. For example, a plan might be generated for repairing a piece of equipment by using a single arm or several arms — under a single controller in either case.

- Planning by Independent Agents. Same as the preceding, except that now each agent creates plans independently. An example of this would be planning by multiple independent robots to assemble a space structure, or carrying out different tasks on the same piece of equipment.

- Geometry-based Spatial Planning. This planning involves reasoning about time and space — e.g., how to assemble/disassemble a piece of equipment, how to construct a space structure, and how to plan a path from one location to another on the space station.

Both scheduling and task planning have been investigated in an attempt to automate the planning process. Scheduling techniques basically search for an optimal ordering of the actions required. Automatic task-planning systems, on the other hand, start with a goal or set of goals, knowledge about the initial situation, facts about the world, and a set of activities that can be carried out. The system will produce a plan containing specific actions involving specific agents that will achieve the desired goals. Some planning research is directed towards developing methods for fully automatic planning, while other research involves interactive planning, in which the decision-making is shared by a person and a computer.

An automatic planning system can be regarded as a type of expert system, but it differs considerably from the expert systems currently in use. In particular, planning systems are concerned mainly with reasoning over time as they search through a space of possible worlds, looking for one in which given goals are satisfied. Thus, these systems use specific representations for describing actions and events, and provide means for determining the effects of combining actions either sequentially or in parallel. Unlike most current expert systems, the methods by which conclusions are drawn are usually non-probabilistic. In addition, the output from a planning system is a detailed plan, not a diagnosis or decision.

B. Space Station Applications

AI-based planning systems will reduce manpower requirements, expedite activity planning for the space station, and, in the long run, produce better plans. Such systems will be needed for scheduling the servicing of both manufacturing operations and onboard experiments. Automatic planning of manipulator movement will be necessary for shifting from teleoperator to autonomous robot systems, while automatic navigational planning will be necessary for mobile robot systems. Automatic planning will also be required for constructing maintenance and repair procedures. Some specific applications of planning systems to the space station are listed below:

- Astronaut and Experiment Scheduling. Schedule crew activities and coordinating experiments.
- Power Distribution. Replan power load distribution as needed.
- Servicing. Plan servicing sequences for equipment maintenance.
- Process Planning. Plan the processing operations of a manufacturing unit.
- Mission Planning. Schedule space station operations and develop mission plans.
- Maintenance Planning. Synthesize procedures for fault diagnosis and repair.
- Adaptive Teleoperation. Plan the submovements required by a robot arm and effectors, based on high-level requirements specified by the human operator.
- Construction. Plan the construction of large space structures
- Autonomous Robots. Plan the movements of an autonomous robot, taking into account the actions of other agents, both human and robotic.

Automated scheduling systems should be available in time for inclusion in the IOC. Planning systems for simple assembly and disassembly, based on the use of CAD data bases, may also be available prior to IOC. This will allow them to be tested in the space station environment, but they probably will not come into common use until several years later. In task planning, interactive systems that allow the astronaut to supervise and optimize the plans being developed offer the

best prospect for the near future. The most sophisticated automatic planning will involve autonomous robots and planning maintenance and repair tasks.

The main advantages of automatic planning, in both the short and long term, are as follows:

- Correctness—Because an automatic planner can keep track of all relationships, it will not "forget" to make all the necessary modifications when adding a new action to the plan. However, its correctness will be dependent on the accuracy of the world model it uses in plan construction.

- Speed—An automatic planner should be able to produce and modify complex plans far more rapidly than a human could hope to do.

- Cost Effectiveness—If an automatic planning system is used extensively, the cost of its development will be quickly repaid compared with the cost of performing the same task manually. Perhaps the greatest cost benefit will be derived from the correctness of the resulting plans.

C. State of the Art

Planning systems have not yet achieved the performance level of current expert systems. Although several research-related planning systems have been reported, they have not yet been used in any operational application Automatic planning of assembly and disassembly is receiving only meager attention. Automatic planning for autonomous robot navigation is a subject in which DARPA is actively interested.

In one type of interactive planning, the system automatically provides plans, and the user is able to indicate alternatives or improvements to the plans. For example, SPOT [1] was designed to assist in planning the movement and landing of aircraft on a carrier. Another related type of interactive system is a plan checker that prepares a critique of a plan that has been prepared by a person, such as the ATTENDING system [2]. ATTENDING prepares a critique of a physician's preoperative plan for managing anesthetics. The program indicates other approaches, and assesses the risks and benefits of the proposed plan. Another plan checker is KNOBS [3], a system for checking aircraft mission plans for the Air Force. Such interactive planning could play an important role in helping astronauts plan spacecraft activities.

The DEVISER program [4] is an expert system for planning and scheduling

actions for an autonomous unmanned spacecraft. In a sensor-scheduling application shown in Figure 45, DEVISER models activities such as moving the sensor platform, recording picture events, changing the filters, starting the recording, and transmitting pictures to earth. Each of these activities requires a certain amount of time that must be taken into account in the scheduling operation to avoid any detrimental interactions. The program generates parallel plans so that an action such as "transmit data" can be carried out at the same time as "move sensor platform." The final plan network resembles a PERT chart.

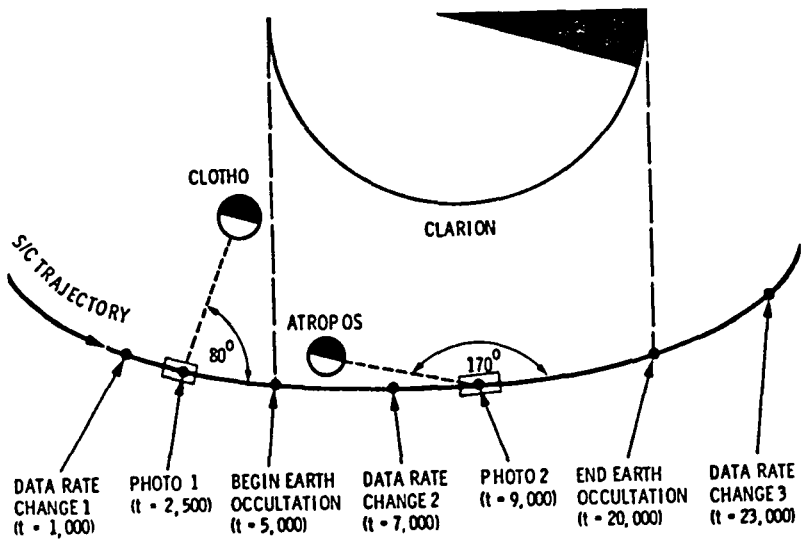


Figure 45: Automated Planning and Scheduling in DEVISER

Recent research (e.g., [5]) has started to investigate the role of knowledge, belief and desire in planning and formulating intentions to act. These concepts become very important when more complex tasks or multiple agents are involved. Other research has studied the representation of actions [6, 7, 8];, the use of constraints [9, 10] and knowledge about resources [10], conditional plans and replanning techniques [11]; and the synthesis of plans involving coordinated parallel activities [12].

D. Research Issues

As background for the research plan to be described later, we indicate the important issues in automatic planning.

1. *Scheduling Pre-specified Actions*

The scheduling problem is an example of planning under known conditions. We are given a set of desired goals and the actions for achieving these goals. Some of the goals may be constrained to begin or end at certain scheduled times; and some of the actions may be carried out in parallel. For each of these actions the necessary conditions are specified, as are the things in the world that are changed by the action.

The critical path method (CPM) and program evaluation and review technique (PERT) are members of a family of scheduling techniques in which one uses a network representation of a proposed project to evaluate a plan for the project. In CPM, the duration of a project is determined by locating the chain of interdependent operations. PERT introduces the concept of uncertainty and allows the user to provide optimistic, pessimistic, and the most probable estimates as to when an activity will take place. Computer-aided critical path scheduling and management control by means of CPM and PERT techniques are in common use, but the activities are given by the user and not planned by the system. For more complex plans, scheduling is accordingly much more complicated. The system needs to consider issues such as possible interactions between subsystems, negation of effects of actions, re-establishing the original preconditions of actions when an action is retracted, and possibilities for deadlock.

2. *Representional Issues*

To develop a plan involving actions, it is necessary to represent the nature and effect of the actions to the degree of detail required by the complexity of the planning situation. The problem of representation lies at the heart of the automatic planning process A good representation is one that captures the way the action changes the world and what conditions are necessary for that action to be applicable to a particular situation. One must somehow indicate the properties of objects in this world that are important to actions, such as movability, danger, and hardness.

It is also necessary to model the environment and, in more complex situations, the belief structures and goals of the agents carrying out the actions Actions can be either deterministic or be characterized by uncertainty concerning their effects. For deterministic actions, the representation must indicate the required conditions (before, during, and after), and the results or effects of that action. For probabilistic actions, the representation must indicate the probabilities of results that can occur. Because planning is a problem solving activity that involves the exploration of alternative hypothesized combinations of actions, a symbolic model of the real world — referred to as a *world model* — is used to enable simple simulations of the situation to be run as the plans are evolved. As with all models, the world models used in problem solving are

abstractions or oversimplifications of the world they model. If a CAD/CAM data base is available, it can be used as a source of descriptions of the *space station world.*

It is also necessary to represent in a computationally effective form the implicit temporal relationships that exist in a situation. These include constraints on possible relations between actions (e.g., before, during, and overlapping) and durational information (e.g., that the performance of a particular action will take between 1 and 5 minutes).

Another important part of the planning process is the utilization of portions of previous plans to solve a current problem. However, one must first solve the difficult problem of how to describe portions of previous plans so that the system will know about their availability for use in the current situation.

Finally, techniques are required for reasoning about spatial relationships among objects such as collisions, fitting of parts, and contact-constrained motions. Spatial representations are necessary to automatically solve such problems as collision avoidance, "grasp" planning, path planning and object joining.

3. *Synthesizing Plans from a Large Set of Actions*

Automatic task planning attempts to develop a plan without manual intervention. If the number of actions is very small and the goals simple, then all combinations of actions can be tried to achieve the goals. However, in a practical problem the combinatorial approach becomes unwieldy, and one must provide techniques that first try out the most "relevant" set of actions, where relevance is indicated by the effect of each action on the world or by heuristic rules that relate goals to actions. One approach is to develop a hierarchy of plans, each serving as a skeleton for the problem solving process at the next level of detail [13]. In this way, complex problems can be reduced to a sequence of much shorter, simpler subproblems. This hierachical planning technique has used in more recent planners, such as NONLIN [14] and SIPE [10].

4. *Execution Monitoring and Plan Updating*

As indicated previously, general planning and reasoning is a very complex problem because of incomplete data resulting from incomplete world knowledge, activities of other agents, and the dynamics of the world. There is often the need to continuously update and modify the plan and its corresponding world model.

An example of this type of planning in the space station might be the following film change scenario carried out by an autonomous vehicle, and described by Boeing [15].

The vehicle might be given the general command to perform a film

change operation at a film-use site. The vehicle would have to plan the following: the routing from its present location to logistic stores, obtaining the film canister, traveling to the film-use site, retrieving the exposed film and inserting a new film cartridge into the canister, depositing the film canister in logistics stores, and returning to its normal location. In each of these tasks, the world may not be in the state expected, e.g., the film cannister may have been moved in logistic stores.

To handle such exigencies, the robot must use its sensors to determine the actual state of the environment at each stage during plan execution. The robot must revise its plans when it encounters unanticipated situations that will affect achievement of its goals. One current system that allows this is SIPE [Wilkins85], which replans when an unanticipated situation arises that invalidates the remainder of the plan.

5. *Planning for Multiple Agents*

All of the previous discussion assumed a single agent carrying out the actions. Future space vehicles will require networks of physically-distributed computer processors or multiple autonomous vehicles or robots. These agents may be under the control of a single controller, a hierarchical control system, or may operate autonomously. The single controller simplifies the situation, but places constraints on the communication links between the agents. In more complex situations, it may be more appropriate to employ distributed systems of specialized intelligent agents, each planning in cooperation with the others to achieve certain goals. For example, it may be desirable to have a variety of agents that specialize in particular areas, such as transporters, manipulators, and system monitors. Each would be autonomous, yet capable of cooperating with others in a flexible manner to accomplish a given task.

Multiagent systems have the advantage of fault tolerance — if one agent cannot achieve a given task, then the task may still be accomplished with the cooperation of others. Such multiagent systems also allow for evolutionary development, because single systems can be added or modified independently of the rest. This is particularly important in areas of fast-changing technology because such rapid changes can quickly lead to intractable integration problems. Distributed systems also allow for high levels of parallel processing, providing a significant gain in efficiency that can be important in time-critical situations.

Independent multiple agents that plan separately may produce plans that are unworkable in practice because of unforeseen interactions (e.g., two agents may deadlock, one waiting for the other to finish some task and vice versa). To avoid such situations, it is necessary to be able to reason about interference

between actions, and to synchronize the activities of the agents.

Many tasks will require the cooperation of agents, either to share information or to provide physical assistance. This requires that agents be able to reason about the beliefs, goals, and capabilities of other agents. Realization of the full benefits of multiagent systems will require research advances in the reasoning abilities of the individual agents and in interagent coordination and communication strategies.

E. Research Funding

The ALV portion of the DARPA SCP has a strong automatic-planning component that emphasizes route planning using stored terrain and other map information. The results of the ALV automatic-planning research may be applicable, in part, to navigational planning for robotic devices that move in a semiautonomous fashion around and in the vicinity of the space station. However, the navigation of robotic devices poses a radically different problem in the space station environment. It requires reasoning about a very dynamic world with moving objects, rather than planning routes in a relatively static domain consisting of varied terrain, enemy positions, and the like.

Since planning in DARPA's SCP concentrates mainly on navigational issues, it will have very little effect on the more general problems of task planning required for the space station. Planning for robotic tasks involving spatial/geometric reasoning (such as repairing a satellite) will benefit from research being conducted under the DARPA/Air Force Intelligent Task Automation program, which deals with automatic assembly. There is also research of this type being done in industry. It may be possible to adapt some of these results to the specialized requirements of NASA.

There is no significant multiagent research being done in the DARPA projects, yet this topic is of critical relevance to many space station tasks in which multiple robots or persons are engaged. Similarly, there is very little in the DARPA projects that is concerned with planning to realize goals or perform general tasks such as repair, construction, or material transport, all of which are essential for space station operations such as satellite servicing, construction of assemblies, OMV operations, and transfer of fuels.

F. Demonstrations

The following demonstrations would confirm the availability of the respective automatic-planning capabilities needed for the space station. They can first be conducted on the ground, then in space. The schedule is shown in Figure 46:

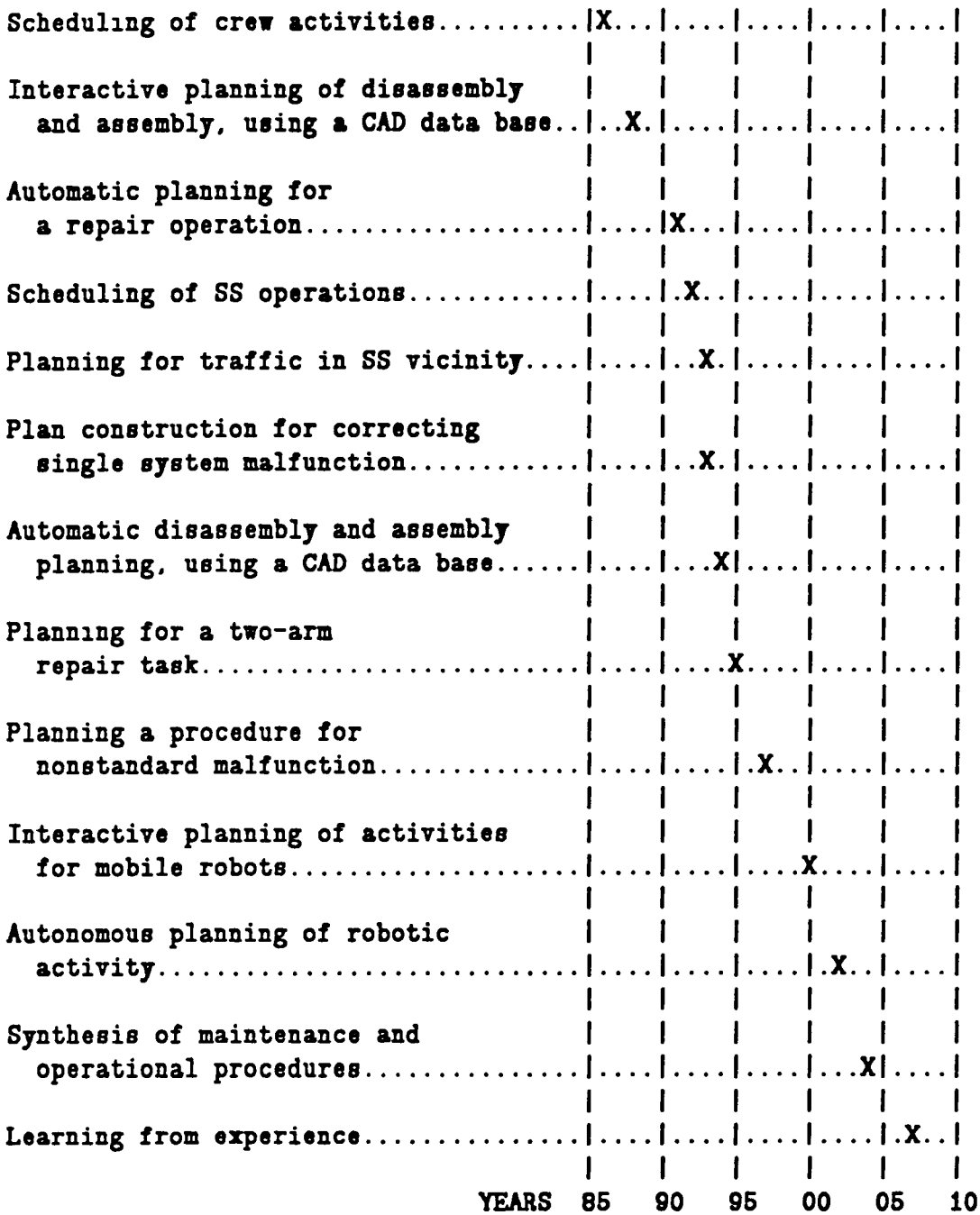


Figure 46: Schedule of Planning Demonstrations

Near-term (1985-1992)

- Interactive scheduling of crew activities.
- Interactive planning of disassembly and assembly, using a CAD data base.
- Automatic planning for a repair operation.

Medium-term (1993-2000)

- Scheduling of space station activities and utilization of resources.
- Planning for traffic in the vicinity of the space station (involving OMV, OTV, EVA, and STS movements).
- Construction plans to correct single subsystem malfunctions using knowledge of system structure and function.
- Automatic disassembly and assembly planning, using a CAD data base.
- Multiple-agent planning for a two-arm repair task.
- Synthesis of procedures for correcting malfunctions not handled by standard procedures.
- Interactive planning of activity for mobile robots.

Long-term (2001-2010)

- Autonomous planning of robotic activity.
- Synthesis of maintenance and operational procedures, taking into account crew safety, integrity of the space station, and expert engineering knowledge.
- Construction of new plans based on analagous previous solutions.

G. Research and Development

The following research and development will be required for the demonstrations. Specific space station planning problems in scheduling, subsystem operation, and robotics should be used as a focus for the research.

Near-term (1985-1992)

- Representation of actions and of resource constraints so as to produce a scheduling system capable of effective user interaction
- Formalisms for representing and reasoning about a wide class of actions and objects.
- Techniques for maintaining the consistency of data bases over time.
- Planning under uncertainty and construction of conditional plans.
- Monitoring plan execution and replanning on plan failure.
- Reasoning about concurrency and synchronization of activities.
- Representation and reasoning about simple objects using geometric information (e.g., contact constraints, attachment points, CAD models, etc.).

Medium-term (1993-2000)

- Representation of actions and of resource constraints so that scheduling systems can be interfaced with automatic-planning systems.
- Complex scheduling involving cooperation, conflict, time-space constraints, and coordination with other schedulers.
- Planning systems capable of synthesizing plans involving iteration, recursion, and other control mechanisms.
- Reasoning about systems of multiple agents involving beliefs of agents, interagent coordination, and communication.
- Plan synthesis for complex tasks requiring reasoning about 3-D spatial relationships among objects and temporal relationships among activities.
- Reasoning about part mating and dissassembly and maneuvering of complex objects in a cluttered environment.

Long-term (2001-2010)

- Reasoning about continuous (rather than discrete) time, and rates of change of system parameters.

- Reasoning about the function and operation of complex physical mechanisms and processes (qualitative reasoning about physcial systems).

- Reasoning by analogy for use in planning systems.

REFERENCES

1. A. Robinson and D. Wilkins, "Man-Machine Cooperation for Action Planning," (Final Report), Menlo Park, California, Artificial Intelligence Center, SRI International, November 1982.

2. P. L. Miller, "ATTENDING: Critiquing a Physician's Management Plan," *IEEE Trans. on Pattern Analysis and Machine Intelligence* (September 1983)

3. C. Engelman, "KNOBS: An Experimental Knowledge-Based Tactical Air Mission Planning System and A Rule-Based Aircraft Indentification Simulation Facility," *Proceedings of the 6 th International Joint Conference on Artificial Intelligence (2 Vols.)*, pp. 247-249, Tokyo (August 1979).

4. S. A. Vere, "Planning in Time: Windows and Durations for Activities and Goals," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, May 1983, *PAMI-5*(3), pp. 246-266.

5. K. Konolige and M. P. Georgeff, "Research on Distributed Artificial Intelligence," (Final Report), Menlo Park, CA, SRI International, 1985.

6. J. F. Allen and J. A. Koomen, "Planning Using a Temporal World Model," *Proceedings of the 8 th International Joint Conference on Artificial Intelligence*, pp. 741-747, Karlsruhe, West Germany (August 1983).

7. P. C. Cheeseman, "A Representation of Time for Automatic Planning," *Proceedings of the International Comference on Robotics*, pp. 513-518, Atlanta, Georgia (March 1984).

8. M. P. Georgeff, "A Theory of Action for Multiagent Planning," *Proceedings of the 4 th National Conference on Artificial Intelligence*, pp. 121-125, Austin, Texas (August 1984).

9. M. Stefik, "Planning with Constraints (MOLGEN: Part 1)," *Artificial Intelligence*, 1981, *16*(2), pp. 111-140.

10. D. E. Wilkins, "Domain-Independent Planning: Representation and Plan Generation," *Artificial Intelligence*, April 1984, *22*(3), pp. 269-301.

11. D. E. Wilkins, "Recovering from Execution Errors in SIPE," (Tech. Rep.), Menlo Park, CA, SRI international, 1985.

12. M. P. Georgeff, "Communication and Interaction in Multi-agent Planning," *Proceedings of the 3 rd National Conference on Artificial Intelligence*, pp 125-129, Washington, D.C. (August 1983).

13. E. D. Sacerdoti, *A Structure for Plans and Behavior,* Elsevier Computer Science Library, New York, N.Y., 1977. Artificial Intelligence Series.

14. A. Tate, "Goal Structure - Capturing the Intent of Plans," *Proceeedings of 6th European Conference on Artificial Intelligence*, pp. 273-276 (1984).

15. "SSAS.Operator System Interface," (Final Report), Seattle, WA, Boeing Aerospace, November 1984. Concept Definition and Technology Assessment.

IX SPACE STATION INFORMATION SYSTEM

A. Introduction

The SSIS* provides data management, data processing process control and support of the crew, terrestrial users and space station subsystems. It also is responsible for maintaining the basic integrity of the space station platform, e.g., in stability, power management, life support and communication. The SSIS has central responsibility for performance, reliability and automation, as indicated in the NASA RFP for preliminary design of the space station. There is a significant need for automation in the astronaut's use of the SSIS itself.

The stringent requirements for reliability and performance place new demands on computer technology. New hardware and software techniques as well as innovative architectural approaches are necessary to accommodate the great variety of kinds of computing functions—many with very high performance demands—and to provide for system growth and evolution. Automated control of the system will require intelligent interfaces to relieve the crew from the burden of process integration; the hostile natural environment requires an extraordinary capability for fault tolerance and recovery from major interruptions and losses of computing resources; the astronaut's need for rapid, unpredictable access to masses of data requires a new methodology for integrating and retrieving such information whenever necessary and without delay; and demands for security and data privacy must be met without seriously degrading performance. To some extent, techniques exist to satisfy most of the individual requirements, but the integration of techniques to meet all the requirements with reasonable efficiency and allowing for future technical growth is a very great challenge indeed.

In this chapter, we identify requirements and current technology challenges for the SSIS. We then discuss various relevant architectural concepts, including hierarchically distributed processing, intelligent-system assistance for user support and maintenance, and integrated data-base management and display. In the concluding section, we identify specific design features for the IOC configuration and technical issues that justify long-range research. In three appendices we consider the role of hierarchical design for distributed systems, provide a more detailed assessment of the security requirements, and discuss trends in storage media.

---

*Also known as the Data Management System (DMS) or the Space Station Data System (SSDS)

B. SSIS Requirements and Architecture

We review here requirements for computing service in terms of SSIS functions and of attributes pertaining to robustness and evolvability, and then examine the controversial issue of overall architecture for the SSIS. The primary question is what architectural scheme will allow reasonable compromises to be made among the many conflicting criteria and still permit orderly growth and evolution?

1. *Requirements for Computational Support*

The following is a review of the computing service that the SSIS must support:

Support for Man-Machine Cooperation. The SSIS must be compatible with ground mission control as well as astronaut-centered, providing the astronauts with effective machine interfaces for decision-making and control and a high level of machine intelligence to extend their reasoning and command capabilities. Astronauts will require access to information at all levels of the system to properly manage the great variety of complex station processes. We envision information-intensive displays, with much more information available in a single view than currently. Where possible, information should be represented graphically, including graphs, diagrams, and pictures, in combination with text. Scenes of physical systems should be generated dynamically and logical processing should be provided to give real-time checking of constraints that apply to the task being performed. Architectural support will be needed in the form of high-bandwidth, rapid access to all system data, powerful, general-purpose processing and data buffering for display formatting and control, and special-purpose processors. Current contractor studies [1] address command mode and device characteristics; equally important is the more abstract question of how information will be represented so that it will be relevant to actual problem solving, and so that information of different types and originating in different subsystems may be presented to the astronaut in a consistent manner.

Significant Computing Power. The wide range of applications (scientific, graphic displays, intelligent interfaces, maintenance) will require significant computing power. Specific requirements will exist for throughput, response speed, and satisfaction of real-time constraints. For some applications the required performance is achievable with standard uniprocessors. Higher levels of concurrency (in the order of tens of processes) are available from distributed processing systems but only in the form of disjoint processing. Levels of concurrency on the order of hundreds or thousands will require new architectures, such as dataflow, for general purpose computing; symbolic processors, for artificial intelligence computations; highly parallel computers, for well-structured problems (e.g., image processing); relational machines, for data base applications; vector

processors, for scientific computation, systolic arrays, for fast algorithm execution, and various signal-processing machines.

Process-Control Automation. Individual control functions for the many onboard processes will be similar to the common earth-based process-control functions, but the number of systems and the complexity of their interactions will demand new levels of automation at the human interface, a higher order of parallel processing, and greatly improved techniques for safe design [2].

Data-Base Management. The SSIS will manage a very large volume of data, perhaps on the order of several terabytes, for dynamic use and archiving [1]. Special techniques will be needed to handle the volume of data, for rapid and logically flexible access to it through a hierarchy of storage devices, and to maintain consistency of the data base when multiple copies are maintained and distributed to provide protection from system crashes. In addition to the problem of data management, we must integrate the models used to represent data. For example, to support the astronaut in real-time problem solving, the data system should be able to derive any complex property about the state of the space station that the astronaut may demand. This requires appropriate data structures and representations, compatible networking of all of the constituent data bases, and the use of a powerful query language to permit the retrieval of information (especially in emergencies) in combinations not always previously anticipated. These requirements imply an intelligent data-base management system that spans all SSIS data. An important goal for the SSIS is an intelligent relational data base that integrates all onboard data, including space station design information (in support of onboard maintenance), as well as operational data.

Signal Processing. A major computational burden in the SSIS is the processing of data from imaging devices, vocoders, radars, and the like, in which the data rate is high and the processing algorithms may be very complex. Special-purpose computers will be required that achieve exceptional performance through the use of extensive parallelism, easy programmability, and the inclusion of symbolic processing within the same computers.

Support for Artificial Intelligence (AI) Computations. As indicated in other chapters, expert systems will have an important role in the space station, e.g., for planning, for scheduling, and for servicing of certain robot functions. There is a corresponding need for such techniques within the SSIS itself to provide intelligent extension of system functions, such as maintenance, security, and human command. Support for this class of computing is best realized by new kinds of computers for symbol-manipulation. Later we recommend an AI-based capability, the "Astronaut's Assistant," to aid the astronaut in using the SSIS.

2. *Service Attributes*

In meeting the requirements for computing service, the SSIS must also meet demands for robustness, programmability and evolvability, as follows:

Fault Tolerance and Recovery. In the event of component failure, service must be maintained and critical data must be protected automatically, with a minimum of human intervention. The present state of the art in fault-tolerant computing cannot yet cope with the multiple simultaneous faults and unanticipated fault modes that may be encountered in the harsh environment of space. It is particularly weak in preserving data and process integrity under conditions of major breakdowns.

Security, Privacy, and Integrity. The SSIS will be shared by users who want to protect their data against unauthorized access, modification, or other abuse. Matters of serious concern will be the integrity of the SSIS itself and its vulnerability to possible penetration through such stratagems as covert use of the ground link or the implementation of alien functions in onboard subsystems (Security and data privacy are explicit objectives, as itemized in the NASA Phase B RFP, Section C-4-31).

Accommodation of Emergencies. Emergencies will require creative reaction by the astronaut. For example, in the event of a major loss of computing power, the crew may need to marshal significant computational potential from remaining resources to carry out some critical computation. This capability will require powerful automatic functions for system diagnosis (both algorithmic and heuristic) and well-designed human interfaces so that the crew may deal with problems beyond the scope of the SSIS recovery functions.

Evolvability and Growth. The SSIS will change significantly during its lifetime to meet new requirements and benefit from new technology. Changes will include new subsystems and new kinds of data bases. Changes in scale may call for a degree of expandability available only in distributed systems. Data structures and logical processing initially based on conventional designs may later be supplemented or replaced by AI-based techniques.

Software Development and Modifications. The cost of space-qualified software may well be a significant portion of the space station program's budget. Major advances in software technology will be essential—e.g., very-high-level languages and intelligent software workbenches, that can drastically reduce the expense of ground-based software development. Software techniques of a different kind, such as application-specific program generation, will also be needed to support onboard programming.

Amenability to VLSI realization. The SSIS will depend on VLSI (and VHSIC) technology to achieve economy of size and power, the potential for high-speed operation, and high intrinsic component reliability. Certain constraints are imposed by this rapidly expanding technology, e.g., the need for standardization of chip type (in face of the ease of special, *ad hoc* chip design) and a need for greatly improved ability to diagnose faults. Advances in VLSI will reduce the overhead cost involved in synchronizing remote processes, and make feasible the evolution from centralized to distributed processing architecture.

There is a significant body of research results that gives promise of initial near-term solutions for each of these attributes. However, we emphasize that no existing information system concept embraces *all* the attributes listed; there is an acute and essential need for a system architecture that addresses all requirements in an integrated manner.

3. *A Physical View of the Computer System*

A recent NASA document [3] describes three generic approaches to the SSIS: centralized, distributed, and hybrid. (An example of a hybrid approach is a two-level system, with a single computer system serving space station-wide functions and a set of machines or local networks serving individual subsystem functions.) The NASA report favors a hybrid approach, as does the following discussion.

Unfortunately, there is a common confusion among physical, functional and control architectures. For example, it is possible to have hierarchical control relations among machines whose communication is provided by a nonhierarchical network. Therefore, it is extremely important to distinguish between physical, control and functional structures. It would be a mistake to design IOC software so that it exploits particular features of the hardware complement to achieve initial optimal economy and performance, possibly sacrificing future system evolvability. Available software mechanisms would simplify using the same software on different physical configurations; examples include message-based module communication; centralized directories of objects by name, location, and access authorization; and tagging of multiple attribute descriptors to individual data items. It is possible to reconfigure software among different physical structures with relative ease by using such mechanisms, even within strictly centralized hardware configurations. (Communication delays may affect performance, but future higher hardware speeds may compensate for this.)

Thus, our view of the physical organization of the SSIS is a system in of physically distributed computing elements, with some degree of dedicated functionality and control, and organized in several networks and subnetworks. Various forms of hardware redundancy will be employed, e.g., dedicated or pooled

spare processors. Initially, some processors will play central control roles, but it is expected that the balance between central and distributed resources will shift to more distributed processing to achieve higher and more efficient fault tolerance and parallel performance. Figures 47 and 48 show the McDonnell-Douglas SSDS team preliminary view of how the computer resources will be distributed on the space station Five data management systems (DMS) are shown located in the habitability modules and laboratories. Figure 47 shows the physical location of the DMS units, and Figure 48 indicates the tasks carried out by each DMS.

Among the possibilities for the computing elements are conventional general-purpose computers, high-performance special-purpose computers, and "smart" devices. Each of the high-performance computers will be optimized for one of a class of functions: numeric processing, symbolic processing (e.g., for logical deduction), signal processing, or specialized computation (e.g., display generation) Each computing element type could require its own specialized programming language, its specialized operating system, and, perhaps, its own environment for creating and modifying programs. Moreover, each of these elements can have built-in facilities for handling most of its own faults and even for protecting itself against abuses (e.g., the data in a computing element might be encrypted), but some pooling of spare computing resources will be provided that can be shared by several subsystems. Many services will require the integration of different processing types (e.g., logical, numerical, signal processing) that may reside in different processors.

A variety of networks and intercomputer communication protocols are available. Some computing elements will be clustered in local networks, linked to other SSIS networks via special gateways. Some networks may be provided with encryption features to enhance network security. The networks must provide multiple redundant paths among computing elements where extreme reliability in the presence of faults in computing or network elements is an issue. The MDAC and TRW SSDS efforts are currently addressing the many choices in network structure and technology, such as token or packet-based control, star, ring and mesh configurations, wire, and coaxial or fiber media.

The relative ease of adding new computing modules to a modern data communication network gives rise to a potentially serious problem; namely, an uncoordinated growth of machines with unique programming languages, operating systems, file systems, and protocols of use. Individual machines may be justified in terms of unique service (e.g., LISP-based machines) but their uniqueness can greatly impede system-wide measures for reliability management, data management, and security. An approach to this problem is discussed below.

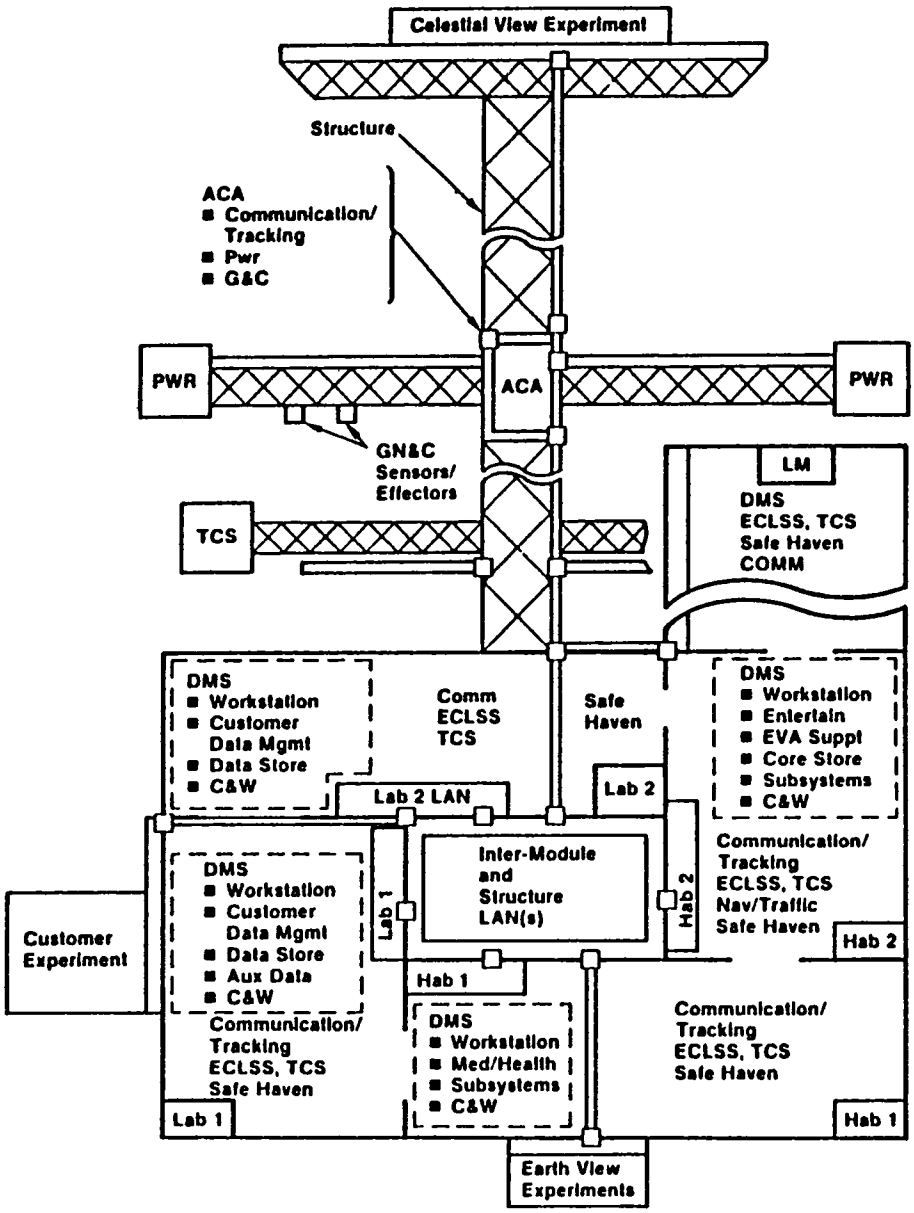


Figure 47: Space Station Data System Onboard Allocation (PRELIMINARY), MDAC SSDS Study

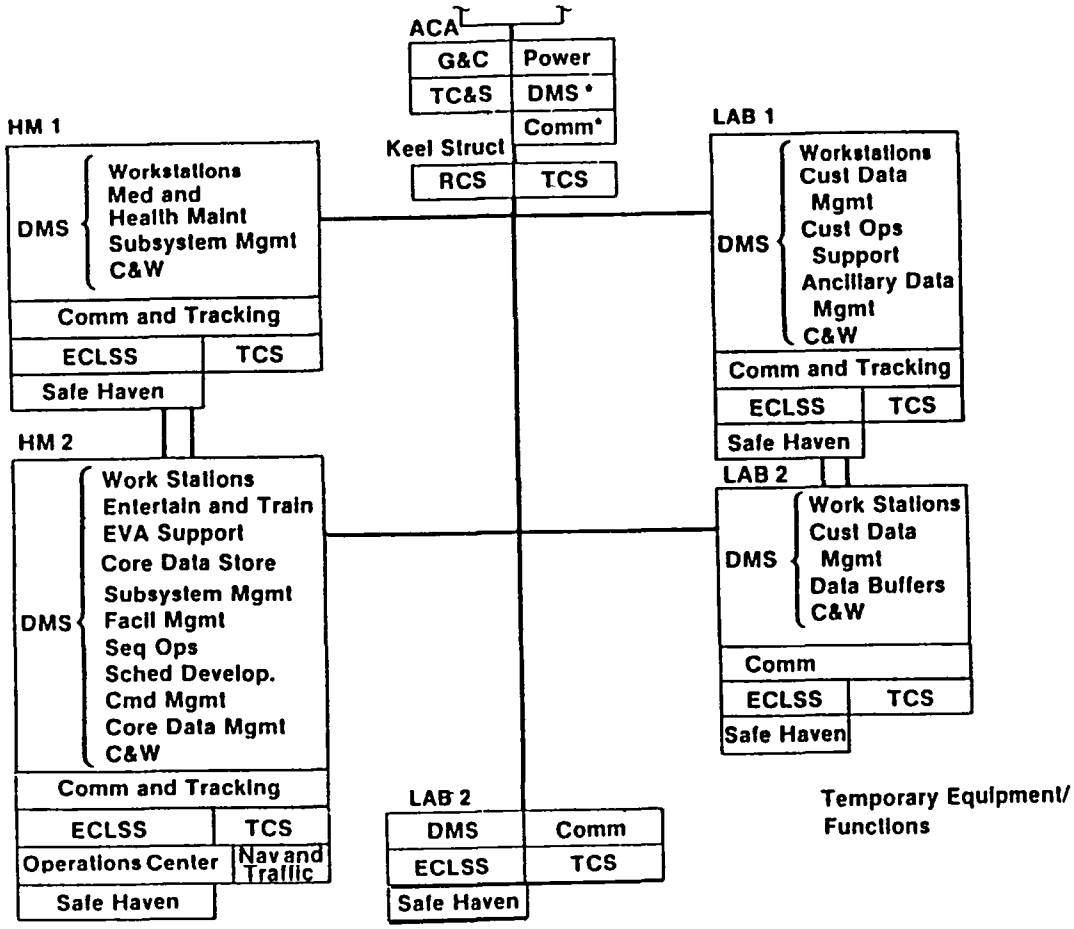


Figure 48: Space Station Data System, Functions of the DMS (PRELIMINARY), MDAC SSDS Study

4. *A Framework for Architectural Design*

We have argued for the mutual independence of hardware and software structures to ease system evolution. Independent evolution of hardware and software is greatly enhanced in logical organizations composed of a hierarchical ordering of functions, ordered from the *abstract* to the *concrete*. This has been dramatically illustrated in the area of network communication by the use of the International Standards Organization multilevel protocol scheme (the "ISO/OSI model"), in which each layer of information transfer functions serves the needs of higher layers and effectively hides details of its own implementation and those of lower levels. Table 4 gives a similar scheme for the SSIS. No standards presently exist for layer functionality in computer systems. Each layer itself will probably require multilevel implementation; for example, the lowest level itself will contain

the equivalent of the OSI/ISO hierarchy, as well as other functions.

Table 4: Logical Organization of the SSIS

Layer	Name	Principal Functions
5	Man-Machine Interface	Multimedia display of information; Instructions in support of application programs
4	Astronaut's Associate	Planning, status monitoring, explaining, Specialized languages in support of multiagent, distributed expert systems
3	Intelligent System Agents	Specialized expert systems for management of the SSIS, e.g., maintenance, configuration management
2	Distributed Operating System	Management of communication among computing elements; Management of redundancy to achieve fault tolerance; Scheduling of resources to optimize performance; Security management
1	Physical Resources	Computing elements and their local operating systems and communication facilities

In the initial SSIS, the layers would use conventional programming, augmented by rudimentary expert systems. As the technology advances, the system can evolve by introducing new capabilities at each layer, e.g., in the area of expert systems. Distributing these functions over multiple levels encourages logical simplicity and efficiency.

A particular benefit of the hierarchical approach is in providing standard interfaces for sharing resources and functions in a distributed system. For example, if all machines in the SSIS assume the same functionality at their file-system levels, it will be easy to have a logically unified system-wide file system, as in the UNIX-United system.

C. Technology Challenges for the SSIS

In this section we examine several technical areas in computer technology that are crucial to the SSIS, evaluating present technology relative to SSIS requirements, and indicating specific targets for technology development. The following areas of technology are addressed: intelligent-system astronaut support, distributed processing, data-base technology, fault diagnosis and fault tolerance, security, and software development.

1. *Intelligent-System Support for the Astronauts*

A major function of the information system will be to relieve the astronaut of tasks that are normally performed by human controllers and decision makers in real-time systems, but are relatively routine and well understood. This section discusses use of machine intelligence techniques for this objective at the human-machine interface and at lower system levels. The techniques are embodied as machine intelligent "assistants" of various kinds.

a The Astronaut's Assistant

Expanding the concept of expert assistance described briefly in Chapter VIII, we envision that the human-machine interface will have the character of an imaginary human assistant that serves the astronaut in all interactions with the SSIS. We will employ an anthropomorphic model and refer to this as **The Astronaut's Assistant**. (There are various current research studies of this notion, e.g., in the areas of programming, aircraft piloting and medical diagnosis). The assistant will act in dialogue mode and, where appropriate, take the initiative to raise issues for the astronaut's attention. The astronaut will give directions for tasks to be undertaken by the system and the assistant will report on their progress.

Given this shared responsibility, part of the interaction between astronaut and assistant will involve development and confirmation of mutual trust. That is, the astronaut will require evidence that the system understands the instructions and is carrying them out properly, and the system will require evidence that an instruction has been carefully considered before undertaking possibly irreversible actions. In defining a task, the assistant will help the astronaut to refer to system objects and define system actions in a precise and natural way. The notion of an astronaut's assistant can be extended to the system as a whole by providing an assistant that is organized in several levels of a hierarchy of responsibilities for each major subsystem. It might be useful to make those assistants visible to the astronaut, and let each have a language of discourse appropriate to its domain. Even if they are not visible to the astronauts, some localization of responsibility for goal-directed control would be helpful to support intelligent cooperative behavior between subsystems, to simplify the programming of intelligent behavior, and to support system reliability.

The notion of a machine-intelligent assistant also can be extended to address system functions for which heuristic extensions (e.g., use of expert knowledge) may be useful in extending the scope and power of conventionally designed system functions, such as fault tolerance, security and dynamic configuration management. Such uses will be proposed in the following section.

b. The Programmer's Assistant

To achieve a high degree of independence from the ground, provision for onboard software development should be made. Applications may include control of experiments, display control, and maintenance procedures. The use of an intelligent agent, acting as a **Programmer's Assistant** would help the human programmer by finding appropriate library packages, assuring consistency and safety within the proposed operational software context, and displaying expected behavior. Use of very-high-level languages and extensive use of graphic input for program specification will further simplify the programming task. This objective is discussed further in the section on software.

c. The Maintenance Assistant

Present techniques in fault-tolerant design will no doubt be employed for individual elements of the information system and for the networks in which they are embedded. Given adequate levels of redundancy, current art provides very good coverage for a significant class of faults, that is, hardware malfunctions that are permanent, few in number, and uncorrelated. Other faults, for which current techniques offer only poor coverage, occur less frequently but have been responsible for a significant number of system failures; these include hardware malfunctions that are transient, or substantial in number and correlated, design errors in hardware or software, and combinations of fault types. Dealing with permanent hardware faults usually is based on some diagnostic algorithm whose coverage of fault occurrences within the class can be predicted with reasonable accuracy. Coverage by those algorithms for faults of the other class is much poorer (where there is any coverage at all), and harder to estimate. Problems of the second class usually require human diagnosis.

Recent research in expert systems has produced encouraging results in computer-system diagnosis. Diagnostic rules for recognizing faults and for evaluating fault-diagnosis strategies have been applied with some success. Therefore, we suggest that a heuristic-based expert system, the **Maintenance Assistant**, be incorporated as a functional level intermediate between the subsystem fault tolerance functions and a human interface, to be used by astronauts when automatic techniques fail. The responsibility of the **Maintenance Assistant** will be to monitor progress of subsystem fault tolerance activities; attempt to diagnose and recover from unusual subsystem and system faults; report and explain unresolved system errors to the astronaut; and perform

diagnostic experiments and recovery actions under astronaut control. The scope of this maintenance aid is concerned only with faults in the information system. Similar interface services and schemes for analysis of diagnostic progress may be found in expert-system aids for space station subsystem maintenance, and should be shared.

d. The Security Manager's Assistant

Given the great number and variety of participants in space station activities, on the station and on earth, there is reason for concern about the possibility of improper access to the system itself and to subsystem data and services Deliberate or accidental improper access can have serious operational consequences, such as damage to data files, denial of computing service, or release of sensitive information. Sources of improper behavior may be individual operators on the station or on earth, and programs that have been embedded in subsystems.

We envision the use of advanced techniques of access security within the space station information system. These techniques provide effective mechanisms for controlling all intersystem data accesses, together with comprehensive formal procedures for individual access. In general, such techniques must be incorporated within the basic system design and cannot be added to an existing design. As in the case of maintenance, there is a role for a heuristic extension to algorithmic techniques, in the form of a **Security Manager's Assistant**. For example, attempts at improper access might be detected by interpreting patterns of user behavior according to heuristically derived rules. There may also be a role for 'intelligent' dialog for the purpose of authenticating the identity of a user.

2. *Distributed Processing*

Distributed-processing architecture offers very attractive SSIS benefits such as reliability, ease of growth, and parallel processing. We noted in the discussion on architecture that the early SSIS will probably be a hybrid of centralized and distributed processing, and emphasized the desirability of modular software design and software-implemented control mechanisms. This would allow the software to make a "transparent" transition from centralized to distributed processing configurations as the technology to support higher degrees of distribution becomes available. We examine in greater detail the technical challenges of distributed processing architecture, discuss the requirements that must be satisfied by a distributed-processing SSIS, review the shortcomings of present art, and suggest specific directions for research to overcome the most critical shortcomings. We review requirements in (1) computing functionality, (2) reliability and security, and (3) evolutionary growth, as they apply to the use of distributed processing in the SSIS.

a. Distributed Computing Functionality

As indicated in the preceding section, the SSIS will have to perform a variety of types of computation, including scientific, symbolic, data base, and signal-processing. It will incorporate computers ranging in size from small microprocessors to supercomputers and will have to support very high internal and external data rates and varied real-time processing objectives. The quantity of internal state information that must be protected under fault conditions will range widely, from a negligible amount, in the case of simple process control, to considerable in scientific computing and maintenance. The physical distribution of physical space station subsystems, as well as sensors and actuators, makes it natural to distribute the processing elements of the SSIS.

Most subsystem processing will be invisible to other subsystems, but some space station operations will span several subsystems, and will require the coordination of highly complex computing processes in many parts of the SSIS. Such efforts can place a heavy time and cognitive burden on crew members. Therefore, a key issue in SSIS design is the level of automation that is provided to the crew for the control and integration of those processes.

b. Distributed System Reliability and Security

Space station operations will require extremely high levels of reliability and security under harsh environmental conditions and the absence of standard maintenance. Reliability has many facets, including probability of correct function for a defined period or average availability of service of an operation; the probability of recovery and recovery time for minor, localized faults or for major system breakdowns; and the gracefulness of degradation when full service is impossible. Security (protection of a process from accidental or intentional interference with normal operation) has also been recognized as a meaningful and important issue.

Distributed processing has intrinsic benefits for reliable and secure design, including: (1) enhanced physical, electrical and logical isolation of faults, (2) convenience of configuration for redundant computing resources, and (3) well-defined and protectable constraints on information flow. These benefits may be realized only by very careful design, because the mechanisms that provide for protection against faults are themselves subject to failure, and careless use of redundancy may seriously reduce normal computing performance.

Another significant reliability requirement is the assurance that chosen computations can meet deadlines in the presence of unusual computing loads and system faults [41983]. A related requirement is that a specified sequence of options must be followed without deviation for critical and perhaps irreversible system actions. Combinations of timing and sequentiality constraints often

appear in the form of *safety* constraints [5].

c. Distributed System Growth and Evolution

The SSIS must grow and evolve over the SS lifetime to meet changing computational requirements and to exploit advances in computing technology. We distinguish between *growth* and *evolution* as follows: (1) *growth* is an increase in size, number or speed of a computing activity that remains within the organizing ability of a given architectural concept, and (2) *evolution* is a modification in architectural concept to accommodate new forms of computing activity or radical changes in scale. Examples of *growth* include more and faster processors, more and larger files and more user functions. Historical examples of *evolution* (perhaps extreme) are the changes from batch to interactive modes and from centralized to distributed processing. Possibilities might include a growing interest in symbolic processing or systolic processing, optical computing, and a quantitative leap in computing scale, e.g., a one-hundred-fold increase in the number of processors.

Distributed processing offers benefits for growth and for evolution by providing uniform physical and logical techniques for interconnecting diverse processing activities. As evident from the existence of large, multilayer, multilinked computer networks (e.g., Arpanet/Local nets), these interconnection techniques can cover an enormous range in number and types of computing activities. The next section discusses the severe performance penalties and logical constraints that presetnly accompany this flexibility.

d. A Framework for Development in Distributed Processing

Current distributed processing systems employ extremely simple forms of distributed processing, such as message passing or remote data storage and retrieval [6], and therefore fully distributed processing for complex, station-wide operations is presently far from supportable. This section presents a view of present and future capabilities in terms of a progression of complexity for multiple interacting processes.

Table 5 gives a hierarchical ordering of distributed processing capabilities based on degree of automation and integration of SSIS computing. The lowest level provides simple communication of data, and the highest provides highly parallel processing activity to support unified mission functions. Level 1, consists simply of the transfer of information between remote facilities, without reference to any computational activity. The next higher level, Level 2, provides interaction between an active, individual computing process and a passive resource, such as a store or an input-output channel. In Level 3, several computing processes may share common resources; they act independently, but they operate within a logically unified data domain.

Table 5: An Automation Hierarchy for Distributed Processing

Level	Level Name	Level Description
6	Definitional Parallelism	Unified computation with implicit parallelism, e.g., Dataflow, Functional programs
5	Programmed Parallelism	Unified computation with explicit parallelism, e.g., Multi-process programs
4	Interactive Processing	Loosely-coupled, cooperating processes
3	Concurrent Processing	Independent computations within a unified data domain, e.g., UNIX United
2	Resource Access	Process control of shared resource—e.g., stores, IO
1	Data Communication	Transfer of data from sources to destinations, e.g., using the ISO multi-layer protocols

In Levels 1, 2, and 3, the purpose of a computation is determined only by the initiating process, a space station subsystem or an astronaut's command. In the higher levels, there is some degree of sharing of purpose among the set of processes active at the level, and hence a higher degree of automation. For example, in Level 4, a complex system task (such as responding to an astronaut's request for information) may be partitioned over a set of cooperating processes such as data processing, retrieval, and data display.

Processes at Levels 5 and 6 support a more unified set of computational goals (or equivalently, they serve a single, ultimate user) Both levels employ parallel processing to maximize performance; at Level 5, parallelism must be programmed explicitly, while at Level 6, parallel activity is derived dynamically from the program text and the state of the computation.

Capabilities at Level 5 and 6 are primarily at the research stage. Several new products at Level 3 and 4 are emerging from the research stage (e.g , UNIX-United and Ports), but almost all current distributed processing systems are at Level 2, and—in general—even those products are deficient with regard to real-time, security, and parallel processing. Common Level 2 services include (1) remote sharing of printers and file servers and (2) transaction processing between workstations and main processors. They are quite useful in offices and other organizations that provide services to individuals, but do not provide for the management of computations that support interacting physical processes, as in an automatic laboratory or factory. Level 1 is well covered in present technology.

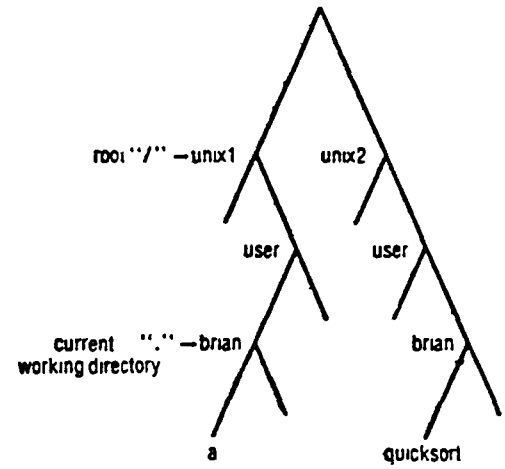
This state of affairs is very unsatisfactory with regard to SSIS goals of automation and growth, which are supported only by the upper three levels.

We are unaware of any distributed system concept that addresses all of the desired attributes, most notably the support for parallel processing, hardware reliability, software reliability, security, and evolvability. However, it is instructive to consider the UNIX-United system developed at the University of Newcastle upon Tyne, England that has some of the needed attributes. In the UNIX-United system shown in Figure 49 (a Level-3 system in terms of the hierarchy presented), objects are referenced through a tree-organized file system, in which the objects can reside in the computer in question or in any other computer in the network. A special function, called the *Newcastle Connection*, intercepts the access and directs it to the machine that holds the object. The logical mechanism for accessing a machine over the network is called the *remote procedure call.* It is clear that a first step towards meeting the SSIS requirement for evolvability is achieved through the Newcastle Connection· computers can be added to the configuration and upgrades to faster computers are possible as long as all of the computers run UNIX. It is not entirely clear if changes can be made while the system is in operation, but this question seems to be easily resolvable.

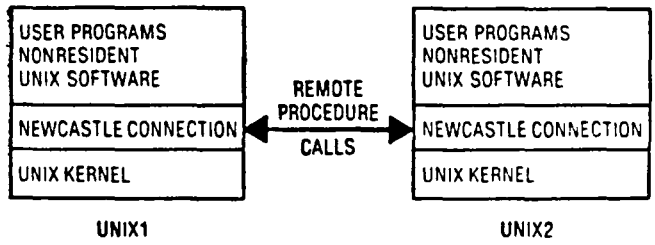
The UNIX-United approach addresses only part of the distributed processing problem. New approaches are needed to meet requirements for parallel processing, fault tolerance, data recovery, survivability, and security in distributed systems.

e. Technical Goals and Problems in Distributed Processing

Achievement of higher levels of parallelism and automation in distributed processing will require mutually consistent and supportive solutions to many difficult problems. A representative set of significant problems is listed here. Several themes, notably fault tolerance, security, and data-base organization, are treated as distinct problem areas in later subsections.



The naming structure of a simple Unix United system



The Newcastle Connection

Figure 49: The UNIX-United System and the Newcastle Connection

Problems of Fault Tolerance

- How can continuity of control and preservation of critical data be ensured when processors fail and when network connectivity is broken?
- How can data replicated in multiple locations be kept consistent with low overhead cost?
- How can the design provide rapid recovery for minor errors and maximum preservation of resources under major error conditions?
- How can redundant hardware and software resources be allocated to enhance reliability according to degree of criticality of the task?
- How can specified time and sequentiality constraints be correctly honored for safety-critical computations, under conditions of high load or system failures?

Problems of Process Integration

- How can rapid name service (e.g., transformation of logical names

into physical addresses) be provided for a continually growing set of relocatable system objects?

- What programming-language constructs and architectural support are appropriate for highly concurrent, dynamic (run-time determined), distributed processes?

- How can the architecture support dynamic relocation of data, programs, and computation site to meet large changes in computational load or system breakdown?

Problems of Security

- Can adequate security mechanisms be incorporated appropriately within the distributed system architecture?

- What mechanisms are appropriate for isolating security-sensitive sections of the SSIS?

- Can a hierarchy of security criticality be achieved, e.g., through identification of a security kernel?

- Can the UNIX-United approach be used effectively (although not necessarily supporting multilevel security)?

Individually, the problems listed constitute substantial research challenges. For example, only one problem, the maintenance of consistency among multiple copies of data under general fault conditions, a Level 2 issue, has accounted for an enormous research literature. One reason for the difficulty of that problem is that the overhead cost for safe solutions is large and the more economical schemes have complex hazards that are difficult to avoid.

Little has been done to develop an architecture that combines sets of the listed objectives, even for such seemingly related objectives as fault tolerance, security and time-constrained processing. Simultaneous treatment of the full list seems to be a too ambitious undertaking, especially considering the overhead burdens of assuring correct performance under fault conditions. Nevertheless, we believe that a sound approach exists for an integrated pursuit of a substantial set of technical advances. In the following sections, we discuss some important design principles to guide new technology developments, and propose a specific strategy for development

f. Principles of Distributed System Design

The following principles of good system design are beginning to be understood, and will be of particular importance to the SSIS. Their application would greatly facilitate the evolution from centralized toward distributed processing, as discussed above.

Principle of Fail-Safe Operation. Akin to Asimov's laws of robotics, the system must remain safe for its users, and for the environment it controls, even when performance is degraded and security is violated.

Principle of Judicious Modularity. Modules should be relatively independent from one another, and should be sharply constrained as to their allowable side-effects on other modules, but excessive modularization should be avoided. Several subprinciples follow:

- Principle of Hierarchical Ordering for Criticality and Functionality—There should be a clear order of dependence that relates to both functionality *and* criticality. The most critical functions should be contained in the soundest modules of the system (e.g., most secure and most reliable), and the lowest layers should never depend upon higher layers for their correct operation. The critical mechanisms are thereby isolated from the less critical ones.* *Thus, the criticality orderings should impose constraints on the design.*

- Principle of Module Encapsulation (vertical decoupling)—Even without a hierarchically decoupled design, the interface to each module, and its implementation, should isolate all data and internal knowledge that need not be visible to users of the interface.

- Principle of Communication Encapsulation (horizontal decoupling, particularly in distributed systems)—Components that may operate concurrently should have relative independence from one another, except through carefully constrained communications, such as a pipe or interprocess communication link. As in the notion of module encapsulation, these communications should isolate distributed components and not permit one component to

*See Reference [7] for a discussion of various relations with which it is meaningful to talk about hierarchies. The point here is that two different notions of hierarchical ordering must be effectively combined into a single notion, respecting both design integrity and criticality.

directly alter another.

- Principle of Separation of Privilege—Use and system privileges should be conferred only where needed. Thus, the critical mechanisms may be accorded greater privilege than the lesser critical ones

- Principle of Communications that Preserve Criticality—The communications within the distributed system must preserve the orders of criticality of each component. For example, for any given measure of trust, no untrusted communication or trusted communication from an untrusted component should be able to undermine the trust of a given component.

We have emphasized the need for a computer architecture that will readily accommodate various emerging new technologies and new techniques. Essential to this concept will be the development of a carefully organized strategy for gracefully introducing new technology into the space station, *on the fly*. This strategy must be particularly resilient with respect to fault tolerance and security, since small changes in the system design, system code, or even only in the system configuration can have potentially disastrous consequences on system behavior. Thus, judicious configuration control as well as great care in making system changes and anticipating their possible effects are essential. Because we anticipate programming and program modification, suitable environments for programming are essential. The use of precisely defined interfaces at each layer of the design throughout the constituent systems and networks, together with the notion of well-hidden implementations of the abstractions (abstract data-type encapsulation) can aid greatly in permitting both small evolutionary changes and large revolutionary changes. The hierarchy then becomes the basis for achieving evolution, fault tolerance, *and* security: in each case, the most critical functions are isolated at the lowest layers of the hierarchy, the next most critical outside of that, and so on. Thus, a philosophy is recommended in which the conventional notions of a security kernel and a fault-tolerance kernel are merged into a single kernel, and significant layers of hierarchical structure are established outside of that kernel. Such an approach appears vital to an effort given the magnitude of the SSIS.

An example of a design approach that incorporates these principles is SRI's Hierarchical Development Methodology [8], which has been used in the development of various real computer systems for both security and fault tolerance, and which supports hierarchical decomposition of system designs to facilitate development and evolutionary change. The notion of a collinear hierarchy that enhances the design of life-critical distributed systems that are

secure and fault tolerance as well is considered in Appendix D, as a further step in this direction.

Several extensions to the distributed processor concept should be considered, to address the other fundamental attributes we see as essential It appears that a *hardware fault tolerance* layer can be placed above the Newcastle Connection to manage the interaction of two or more computers working together to respond to hardware faults. Similarly, there could be a *security* layer that would intercept all accesses to decide if they would result in any flows in violation of a security policy. Furthermore, there could be a *software fault tolerance* layer that would manage different approaches to software reliability (see below).

One additional issue of concern to the distributed system concept is the role of Ada, which is likely to be the language used for the general-purpose computers on the space station.* It would be desirable to use Ada for *all* programming tasks—system software and applications; as discussed below, the use of a standard high-level language clearly facilitates evolvability and portability of software. Moreover, it would be desirable to use the Ada run-time system where possible, since it provides many of the features required of an operating system. However, it will be necessary to enhance Ada with facilities to handle security, hardware fault tolerance, and software fault tolerance.

g. A Kernel Approach to an Evolving, High-Performance Distributed System

We suggest an approach to organizing the SSIS so that it can evolve towards a high-performance, high-quality system of the kind described. At the beginning of its evolution only a portion of the system would exhibit to a high degree all of the desired properties such as fault tolerance, dynamic process management, security and safety support, generalized name management, and some performance goals might be compromised. Such high-quality, critical subsystems are sometimes called *kernels*.

A second use for a kernel is as an ultra-reliable, low- or medium-performance controller for extremely critical functions, such as support for space station integrity (power, stability, life support, etc.) In this concept, the kernel would provide critical space station computations, and services that support the integrity of the SSIS itself, e.g., reliable name management and trustworthy recovery from major failures.

As technology advances, the overhead costs of the high-quality solutions

*Ada might not be be the optimal language for programming the highly parallel computers

used in the kernel will be reduced so that it can be economically expanded to occupy a growing fraction of the SSIS. We envision a kernel with the following technical characteristics:

- Kernel functions are designed so they can be distributed throughout the SSIS.
- Hierarchically organized fault tolerance and recovery (each level attempts to serve its own errors and the unresolved errors of lower levels)
- Self-documenting (tagging) of all data transfers to allow explicit control for multiple objectives, including security, correct time and sequentiality, version management (for redundant processes), etc.
- Redundant processor and interconnection clusters for extremely high fault tolerance.
- Architectural support for monitoring evaluation of correctness and safety conditions.
- Architectural support (e.g., distributed caches) for dynamic data relocation.
- Architectural and language support for a high-level, data-flow mode of process management.

Intensive work on a design with these characteristics should bear initial fruit as soon as the SSIS design begins, with continuing effects thereafter.

3. *Data-Base Technology*

In this section, we review some of the special problems of data management in the SSIS and indicate areas of technology advancement.

a. Data Management in the SSIS

Space station operations will be very data intensive, both in the amount of data generated and in the amount of data needed to support operations. The following are examples of space station operations that may heavily stress SSIS data management capabilities:

Support of astronaut operations—When performing critical, perhaps irreversible operations, e.g., shuttle docking or the starting up of a high-energy experiment, astronauts will want a display of all information that may be relevant

to the operation. This may require retrieval of data about many space station subsystems To do this effectively, the SSIS should employ a model of how the subsystems interact in that specific operation. The astronaut may have to help to define just what data are relevant, perhaps via some kind of graphic input-output medium. Current data query and retrieval systems place an unnecessarily heavy burden on the user to correctly and completely define the data to be retrieved.

Support of onboard maintenance—Autonomous, onboard maintenance will require (1) a detailed description of the design of space station facilities (a potentially staggering amount of information), (2) a history of occurrences, such as faults and configuration changes, that may affect the system's response to future faults, and (3) good models that allow tracing the possible consequences (and propagations) of faults. These models will be more effective if the representations for the design information and the maintenance history will support some degree of logical inference, e.g., for use by an expert system. Current data-base models do not provide adequate support of this nature.

Use of the data-base for constraint monitoring—Space station subsystems will be heavily constrained in their operations to avoid excessive stress and wear, and to avoid hazardous or wasteful modes of operation. The notion of real-time alerts has been suggested as a way of ensuring full and rapid coverage of such concerns (MDAC op. cit. pg 216). As a way of implementing such alert functions, it would be valuable to embed logical expressions of specific constraints in the data bases that record the state of various subsystem and to ensure that they are evaluated at every change of state of the relevant data-base variables. This approach would supplement the use of centralized routines for constraint monitoring.

Support of error recovery—To recover from minor and major losses of SSIS data without ground intervention, some forms of redundant storage will be necessary. Unfortunately, designs for maintaining consistent and up-to-date copies of data tend to have very high overhead, which can seriously reduce data-base performance. New techniques that may help to improve the trade-off characteristics of those solutions include data approximation and compression, differential stores (tracking only changed state variables and transactions), selective storage based on criticality and variable-rate copying. A special case of the redundant storage problem arises in the exchange of data with ground stations. Significant changes in space station state should be reported to earth, but the choice of frequency and detail of reporting is a significant problem, because in extreme cases, ground stations may play a crucial role in data recovery.

Data-base interfacing and integration. The SSIS will contain a variety of

subsystems, built by many different suppliers. Those suppliers will tend to use internal data models that seem natural to the function of the subsystem, such as trees, lists, abstract graphs, relations, and spatial structures (e.g., maps and diagrams). In most cases the structure of the data base and the internal representation of the data can be hidden from the SSIS. In other cases, data may need to be combined to describe the interaction of several subsystems, and a model will be needed to describe that interaction. Integration of information that is described by different data models and representations is a largely unsolved problem in present data-base art.

b. Summary of Data-Base Technology Needs

The kinds of service described require a substantial advance in the current data base art. Specific technical goals include the following:

- Data retrieval based on problem definition and support for user interaction in data definition.
- Graphic and mixed graphic-text modes of query input and display.
- Data representations that better support the inferential processes of problem-solving systems.
- Online monitoring of critical system data.
- Efficient techniques for data protection and recovery.
- Techniques for integration of multiple data models and representation.

4. *Fault Diagnosis and Fault Tolerance*

In this section, we discuss current capabilities and needs for advanced techniques in fault diagnosis and fault tolerance. The general role for these techniques in the SSIS is discussed above in the section on distributed processing. The importance of this subject is recognized by the major SSDS contractors.* Presently, the art in fault testing and diagnosis and of design for testability lags far behind the rapid growth in device and system complexity.

*For example, TRW SSDS 3rd Quarterly Briefing Foils, pp 4-64 et seq

a. Basic Issues

Despite careful design efforts, many types of faults—transient, intermittent and permanent, hard and soft, and including both hardware failures and design and programming errors—will inevitably occur in the SSIS. Providing a system that can tolerate such faults will require a balanced set of protective measures carefully integrated into the system design at several levels. These measures must provide for high-coverage fault *detection,* through both continuous and periodic checking, to reduce to an acceptable level the likelihood of two or more simultaneous faults that can be very difficult to identify. They must also provide fault *location* (loosely called *fault diagnosis*), to narrow down a detected fault to the smallest replaceable unit (SRU) in which it falls; fault *handling,* in the form of reconfiguration or repair, automatic or manual; and fault *recovery,* to return the system to an acceptable reference point from which correct operation may recommence. All four of these steps in the life cycle of a typical fault must themselves be carried out correctly, even in the presence of the fault they are trying to tolerate. Moreover, the overall design must be carried out with an adequate degree of *fault isolation*—the property that prevents hard or soft faults from propagating from one SRU to another.

These requirements dictate a system design that is

- Judiciously redundant at appropriate system levels.
- Segmented into modules that can replace each other to some degree and that can participate in cross-diagnosis (one or more modules checking another) and self-diagnosis (a module checking itself).
- Supportive of the system software that manages fault detection, location, handling and recovery (e.g., providing adequate time during operations for testing and diagnosis, adequate monitoring of internal data, and adequate access to internal state variables)

Achievement of such fault tolerance design goals, even for a system simpler than the SSIS, requires that the fault tolerance be incorporated into the system design at an early stage—not as an add-on to a nonfault-tolerance design. Many fault-tolerance techniques are available to aid the designer, though they are not often applied in practice. For example, "design for testability" is an old and very valid principle commanding attention again as testing issues approach criticality in VLSI and system design. In addition, various models, analyses and simulations are normally required to *validate* the effectiveness of certain system design steps—for example, to determine the fault coverage of diagnostic test sequences, and to measure the reliability resulting from a trial reconfiguration algorithm.

On the other hand, some useful design principles have emerged:

- For large, heterogeneous systems, economical and effective fault tolerance requires an *hierarchical* design approach, with the fault tolerance distributed over several design levels.

- Fault tolerance requires some degree of *segmentation* of the system into parts (modules) that have information-limited interfaces between them (to provide fault isolation), and with as high a degree of replication (identical modules) as performance, cost and functionality will allow (to provide effective diagnosis, reconfigurability and recoverability).

- Design for testability, graceful degradation, and design verification (correctness, performance, reliability) are essential aspects of fault-tolerance design for large systems.

- Fault-tolerance designs tend to be simple and conservative, with strong emphasis on *organization* and *structure* as opposed to such features as intricacy and minimality.

All of these considerations are directly relevant to the SSIS. They must be used as applicable and extended as needed to achieve space station objectives.

b. State of the Art

A broad variety of fault-tolerance design approaches and techniques are presently available, some of them refined and tested through actual experience, Because of the diverse information processing requirements of the SSIS, most of the techniques are potentially relevant to SSIS design. However, the lack of a single underlying theory of fault-tolerance, or even a unifying methodology, prevents systematic design and prediction of the resultant reliability and availability, except for certain small, special-purpose systems—and certainly not the SSIS. For example, it is not always clear when to use a particular technique, nor how effective it will be, and there are many important gaps in the family of known techniques that prevent universal application in real situations.

More specifically, *fault detection* is fairly well developed. It can ordinarily be accomplished at reasonable cost (in hardware, software, time and performance) and with some choice in techniques: low-level hardware redundancy (5 to 25 percent), data encoding (with error-detecting codes), computational checking (check sums, consistency checks), run-time functional checking (overtime checks), and the like. Moreover, by using a combination of these methods one is not constrained to any particular fault model. Both self-test and cross-test methods

are in common use; their effectiveness is well proven and their costs (including the need for back-up data files for error interpretation) are well known.

The major present efforts in fault detection deal with reduction of the cost of detection circuitry or programs for achieving very high fault *coverage*—the fraction of faults detected, out of those for which detection is desired and intended. The limits of currently known methods arise from the severe demands for high coverage placed upon them by the extreme cost of fault location to accommodate *multiple* faults. That is, the cost of locating and handling two or more faults (perhaps correlated, because of a shared weakness or a common external disturbance) at the same time can be unreasonably high. Thus, it is usually preferable to design the fault-tolerance system to operate within a single-fault limit, but to reduce the fault latency (probability of residual undetected faults) to a very low value by making the fault detection coverage very high. This is done by quickly and thoroughly detecting single faults, followed by accurate fault location and handling. Such action is essential in the portions of the SSIS on which life depends and in any case is critical to mission success.

In other respects, *fault location* techniques are similar to those for fault detection. The same detection techniques are applied somewhat more locally and are augmented with a protected (i.e., fault-tolerant) analysis step to decipher error reports into the identity of the defective SRU. The possibility of multiple, correlated faults is a serious concern in the SSIS. Such fault conditions will be more likely than in ground-based systems because of the harsher environment (cosmic rays, extreme temperatures, etc.). Multiple faults confound fault detection algorithms that assume only single faults. Recent research in expert system techniques has been aimed a this problem (e.g., Genesereth's DART system).

In the SSIS *fault handling* will normally be implemented through reconfiguration, reserving repair and massive manual replacement as a last resort if automatic techniques prove insufficient. Reconfiguration techniques appropriate to a large, integrated system such as the SSIS are available, but they all constrain the design to a minimum degree and type of replication, a minimum SRU size, and other factors tied to performance and computational delay. These constraints will probably not be satisfied in the SSIS. For example, highly dynamic reconfiguration schemes and strategies typically require several nearly identical modules of microprocessor size (or larger), and operating in a repetitive computation cycle with frequent breakpoints. Such a high degree of replication and regularity is surely not necessary for successful and economical reconfiguration, but the requisite techniques have not yet been developed.

An important issue bridging diagnosis and reconfiguration is the need for

rapid discrimination between transient and permanent faults. Misdiagnosis of a transient fault as a permanent fault may disable fault-free hardware or program modules; misdiagnosis of a permanent fault as transient raises the likelihood of latent faults, hence multiple faults, that can exceed the fault-tolerance capacity and cripple the entire system. This issue has particular relevance for the SSIS because the crew will not have the expert judgment needed to resolve ambiguities, and yet will be reluctant to discard computer resources.

The limitations of available *fault recovery* methods are similar in degree, though more dependent upon computation mode than degree of replication. For example, subsystems that operate with a fixed, short computation cycle, such as control systems, automatically provide convenient breakpoints to which return can be made for recovery in the event of an error. *Recovery blocks* can sometimes be employed to provide alternate computation modes to reduce vulnerability to correlated faults. Recovery methods for computation cycles that are long or that involve large amounts of tightly connected hardware (which can occur in signal processing) are presently inadequate.

A problem of special concern in the SSIS is recovery from massive (e g , total system) breakdown. Even with suitable fault isolation between subsystems,* custom design will be required for the SSIS. Techniques for tolerance of single faults require prior design experience with similar large systems, and thus may initially be of limited relevance. More relevant approaches include layered system design with an ultra-reliable kernel for control of reconfiguration, expert-system heuristics for diagnosis, and good graphic support for crew-controlled recovery.

Design validation theory and practice is moving forward, but can presently only be applied to small circuits and programs, and only when these are designed in accordance with particular specification rules. It is unlikely that these methodologies will advance to the point of validating the entire hardware or software of the SSIS before the year 2000—and perhaps not even then. On the other hand, if proper design practices are followed, critical kernels of the SSIS may be capable of validation within the next several years. Actually, these design practices are generally recommended for achieving other desirable system features, such as reliability, security, flexibility, maintainability, and evolvability.

---

*Total isolation of subsystems from one other is likely to be more of a hindrance than a help in this eventuality

c. Expected Future Achievements in Fault Diagnosis

Current trends in fault-tolerance technology are toward larger SRUs; more reconfiguration and less replacement; more software and less hardware redundancy (including less special-purpose hardware); increasing emphasis on design-for-testability (including built-in test, modularity, and distribution of functions over the hardware), instead of trying to create tests after the design is completed; and (as implied above) heavier reliance on hierarchical, structured design methods in which the fault tolerance is distributed over several design levels. Diagnostics are moving toward more functional testing and away from gate-level testing as better design methods permit more thorough specification at upper design levels. Analysis and simulation are also becoming more thorough, with the growing emphasis on computer-aided design methods.

In future years, the pace of industrial work in fault tolerance will increase under the pressure of commercial and government applications requiring high reliability in medium and large special-purpose systems and in VLSI devices. However, research laboratories will continue to work on leading-edge issues (correlated faults, hierarchical recovery, and the like). No totally new approaches for effective and economical fault tolerance have appeared on the horizon, either from theoretical research or from applied work, but new needs continually arise for applying fault tolerance to systems based on new technology, e.g., systolic arrays, optical computing, etc.

For the special objectives of fault testing and fault diagnosis, somewhat more effective diagnostic tools can be expected, mainly from semiconductor manufacturers and the immediate users of LSI and VLSI components, as a result of the existing critical need for improved methods of test-generation, fault simulation, and built-in test within the manufacturing environment. Advances can be expected in design approaches that facilitate testing, but there is no indication that the much-needed breakthrough in this area will occur within the next few years.

d. Research Issues in Fault Tolerance

The following research issues are particularly relevant to SSIS needs in fault-tolerant computing. Specific work on all of these should be undertaken in the context of distributed system architecture.

- Incorporating efficient error logging and monitoring onto the IOC to support the analysis of significant (and perhaps unexpected) fault types and discrimination between transient and permanent faults.

- System design, expert heuristics, and human interface design to

support recovery from massive breakdowns.

- Use of expert knowledge to extend the range of covered fault types (using human knowledge plus results of experience, e.g., via error logs). This can be important, e.g., in discriminating between hardware malfunctions and design mistakes (hardware or software).

- System-level integration of built-in testing and external testing at the chip and board levels.

- Investigation of the feasibility of software fault tolerance (run-time correction of errors resulting from design flaws). Several major schemes (N-version and Recovery Blocks) are under study at this time. Special issues include programming techniques and architectural support, as discussed in the section on software methods.

5. *Computer-Communication Security, Integrity, and Privacy*

As a widely shared, real-time system, the SSIS will face many requirements for security. These include maintaining data privacy, i.e., protecting sensitive system data and user data from being read by unauthorized individuals or programs; maintaining the integrity of the computer systems, i.e., protecting them from tampering; and avoiding unauthorized denial of service, i.e., protecting against intentional or accidental unavailability of critical resources. This subsection reviews the state of the art and indicates directions for improvements to support SSIS objectives.

a State of the Art

Existing computer hardware and operating systems are usually seriously flawed with respect to the enforcement of security and privacy. Only a few operating systems have been able to withstand penetration attacks. Even in environments in which *no* users are allowed to write or install *any* programs, vulnerabilities are known to exist. Unfortunately, the purveyors of computer systems are often not aware of all of these vulnerabilties, or at least do not consider them important. In a critical environment such as the SSIS, the problems of security, privacy, system integrity, data integrity, and prevention of accidental or intentional misuse are very difficult. At present, only two computer systems are known to be reasonably secure (Honeywell's SCOMP and Multics) for critical applications. (The SCOMP has undergone extensive formal verification and withstood penetration attempts. A secure network is provided by the Newcastle Connection. (The Newcastle Connection is a secure distributed system composed of nonsecure component computers. Most other systems (UNIX and

most IBM systems included) are easily penetrated. Presuming that a less secure computer system is used in the SSIS, there may indeed be significant flaws in security in the SSIS. The importance of security in the SSIS has thus far been seriously underestimated. Thus careful encapsulation of an environment is essential if it is to be implemented on a nonsecure computer base.

Most current applications include computers that are intrinsically not secure and that are used in generally unsafe modes of operation. Significant effort is required to reduce the risks, although there are usually simple measures that seal off some of the most obvious risks. One such risk is interference with space station activities by students using the account of a professor who has legitimate interactions with onboard experiments. Isolating payload customers from each other is another. While measures can be taken to reduce these risks, they generally cannot be eliminated.

b. Space station Security Needs

We believe that it is essential in the near future for NASA and its contractors to establish a detailed set of security requirements for the SSIS and SSDS. In general, security is in very difficult if not impossible to achieve unless the system design and underlying hardware are suitable for the desired requirements. If those requirements are not clearly stated in advance, the problems are all the more difficult.

It is common—but exceedingly dangerous—to assume that all authorized users are benign, and even worse to assume that there are no unauthorized users. Given the sensitivity of much of the instrumentation, experimentation, and data onboard, a strongly defensive position must be taken with regard to security. Some of the security-related issues implied here are very deep and require extensive elaboration. To this end, a checklist of potentially relevant issues is found in Appendix E and some preliminary analysis of what the actual requirements might be concerned.

c. Further Research

At noted in discussing distributed systems, the use of architectural notions that can enhance security presents important technology challenges, e.g., the use of hierarchies, security kernels and the secure distributed-system network, UNIX-United. A very important direction is the establishment of a design approach that encompasses system security, data-base security, fault tolerance, life-critical operation, and other requirements within a single design framework. The notion of a collinear hierarchy considered in Appendix D is considered to be an important step in that direction. Some security issues are discussed in that appendix (as well as in Appendix E) that relate security to the other critical design requirements within the hierarchical approach to criticality.

6. *Software Methodology*

In this section, we discuss capabilities and requirements for research in software development and software reliability.

a. Software Development

Software development will be one of the major cost components of the space station program. There are two distinct needs for advanced software technology: (1) ground-based system development for the initial SSIS and on-going development, and (2) onboard software development, for modification, debugging, and *ad hoc* program creation.

The very challenging ground-based software development requirements include economical design and verification, and easy system modification and growth. Unfortunately, in software developments of the size and variety needed for the SSIS, there is a tendency for uncontrolled growth, with the familiar consequences of excessive cost, unreliability, and unmodifiability. This tendency can be countered only by applying modern techniques of software engineering, e.g., abstraction, encapsulation, formal specification, and parameterized design, and by supporting those techniques within modern software development environments. The design of such environments is a major current activity in computer science and deserves substantial support as a space station program.

Software development strategies often concentrate on the choice of an implementation language and its supporting environment.* This is an important issue, but the greater need in software development, especially for control of large system development costs is a methodology (including languages and tools) for system design. Use of implementation languages (Ada, HAL, Modula, and the like) for this purpose leads to harmful complexity and over-specificity. New approaches to system design methodology are discussed in this section.

Onboard software development has challenging requirements of a different nature. Astronauts may be called upon (1) to analyze system software for design errors (both detected and suspected), (2) to install software changes, and (3) to develop software as needed for unanticipated operations. The first two requirements imply the need for software tools that provide detailed and intuitively clear representations of program structure and behavior. Assuming that the crew does not include expert programmers, the third requirement implies a high degree of automation in software production. In general, such automation is available for programming that is done within a very specific domain, e.g., process control. However, there is a need for such program generation that has

---

*For example, TRW SSDS 3rd Quarterly Briefing Foils, pagees 4-19 et seq.

greater domain generality.

The following technical approaches have potential for valuable return for one or both of the software development modes described here.

Very-High-Level Programming Languages—The use of a very high-level programming language is very attractive for SSIS to increase software reliability and system evolvability. Despite recent advances in existing high-level programming languages, those languages are inadequate for the implementation of distributed systems. Additional facilities should be developed to support intermodule and interprocess communication, synchronization, atomicity, scheduling, and message passing.

Application Generators—An application generator can be viewed as a very-high-level language that is oriented to a particular application. Such generators would be extremely useful for onboard SSIS software generation (e.g., for robot programming, control of scientific experiments, and onboard maintenance), and allow a user who is not necessarily an experienced programmer to specialize a program for his particular needs. The availability of such programs for applications of direct concern to the space station (e.g., navigation, flight control, maintenance expert systems) would give the astronaut the freedom to perform some modifications during the mission.

Program Libraries—Sometimes a modification is required beyond the capability of an application generator. One approach is to provide the astronaut with a collection of program components accessible from a program library. Although this idea is not new, only with newly emerging technologies does it seem to be practical; in the past most programming libraries were accessed only by the lender and not by the borrower. Among the technologies that should lead to practical programming libraries are: languages for expressing general, widely usable designs through the use of *parameterized* designs, languages for combining modules into a cohesive program, cataloging and searching techniques, and tools to determine the suitability of a module in a particular situation. Techniques should be developed for building up a library of modules suitable for development of future SSIS software.

Program Transformation—Here we consider situations in which the change required to a program is beyond the capability of an application generator and cannot be effected by merely replacing certain components with others in a library. This kind of change is the most difficult and usually results in additional errors being introduced. Program transformation is a promising technique to manage such major changes. Using program transformation, the problem to be solved is initially expressed in terms of a simple but inefficient program; the

premise that makes this work is that an inefficient program to do a particular task is easier to produce than a more efficient program to do the same task. This initial program is transformed to its final and efficient version through a number of steps (perhaps a great many) that introduce design decisions that preserve the original intent of the program by make it more efficient at each step. The transformations are recorded in machine processible form and, furthermore, are checked by tools to assure that correctness is maintained. Later, when a modification is required, the astronaut merely has to go back to the earliest decision that is to be changed and determine what subsequent decisions might have to be changed. We envision the technique of program transformation as being particularly useful as programs have to be changed to accommodate the replacement of serial computers with parallel computers. NASA should study this and other applications of program transformation for possible development as a practical technique for onboard software development.

b. Software Reliability

A number of advanced techniques for obtaining reliable programs through promise of better *coverage* have been investigated and some have been applied to real systems, although extensive research is still required. Work in this area is continuing; NASA's role might be to concentrate its attention on specializing the techniques for the kind of software that will appear in the SSIS. Some of the most promising advanced techniques are listed here, in approximate order of increasing effectiveness in achieving high reliability (but with increased cost).

- *Inspection* implies careful desk-check review of the code by a team of inspectors. No tools are used.

- *Formal Testing of Code* assures that every part of the program is exercised by at least one test. There is no assurance that the tested code is correct, as the technique does not guarantee that code to handle an important situation is present. There are tools available to assist in this process.

- *Formal Testing of Code with Specifications* combines the technique of "formal testing of code" with the generation of tests from specifications of the system. The technique does help assure that the specifications and the code are in correspondence, but suffers the same limitation of all testing techniques: because testing does not handle every possible input to a program, it cannot guarantee correctness in all situations.

- *Software Fault Tolerance* uses redundancy to detect and handle software errors. In one technique, *N-Version Programming*, three

(or more) different versions of a program (all intended to yield identical results) are run simultaneously and the respective outputs are compared. Any single error, then, is masked; the basic technique employed here is called *forward error recovery.* The reliability of this technique is very dependent on the independence of the different versions; an error in the specification would, of course, be present in all versions. In the other technique, *recovery blocks*, the output of a program is checked using an *acceptance test.* If the check fails, an alternate version of the program is executed using the same input as applied to the original version. The reliability of this technique is very dependent on the ability of the acceptance test to capture the specification of the program.

- *Formal Design Verification* proves mathematically that a given specification satisfies its requirements (e.g., for human safety, security, or fault tolerance).

- *Formal Design-Refinement Verification* demonstrates that successive refinements of a design specification are consistent with one another and sufficiently complete.

- *Formal Code Verification* demonstrates by mathematical proof that a program and its specification are consistent. (Coding and compiling are really the last steps in the design refinement.) Tool support for verification, absolutely essential if the proofs themselves are to be trusted, is under development. At present verification has been applied only to moderate-size systems (less than 10,000 lines of code), but as the tools become available outside the research community, large systems will be verified. The reliability of the technique is primarily limited by the ability of a designer to capture his intent for the system in a specification.

D. Research Funding

Principal support for computer research comes from the DARPA SCP, the expanded supercomputer research of the Department of Energy, and NSF's Supercomputer Access Program. Since DARPA's architecture program comprises the largest single federal project addressed to supercomputing, its activities will be broadest and most crucial. Work in many areas will be sponsored, including underlying technologies, improved design automation tools, fabrication means for both VLSI and broad-level special systems construction, conceptual exploration of

new special general-purpose parallel computing systems, various small-scale prototyping efforts, and a few critical large-scale prototyping efforts.

The most important NASA study relevant to space station computing is the 18-month data-management study, "Space System Data Systems (SSDS)," that began in mid-1984, and carried out in two major contracts. The McDonnell-Douglas team is monitored by NASA GSC, and the TRW team by NASA's Johnson Space Center. In a much smaller effort, the Draper Laboratory is studying a fault-tolerant distributed-processing system for the space station, under a Johnson Space Center contract.

Data management is part of the NASA Space Station Advanced Development Program which focuses generic technologies on space station applications; prototype components are built and integrated into subsystems for demonstration in ground-based test-bed facilities, and flight experiments are conducted using the shuttle. Three mutually supportive technology developments being pursued simultaneously in the data management area: (1) electrooptical components; (2) advanced networks and protocols; and (3) modeling and analysis tools. Some of the activities that will be carried out are

- The current MIL-1553 bus architecture will be superseded by advanced technology for network organizations, topologies, and exchange protocols.

- Data storage technology such as optical, magnetic disk, magnetic bubble memory devices will replace magnetic tape recorders.

- Current redundancy management of online computers will be replaced by advanced fault-tolerant, self-checking computer modules for supervisory and control applications.

- Advanced software systems technology will develop interface and protocol standards to improve information exchange.

- Expert systems will be applied to planning and control of flight operations and automation of individual subsystems.

The data-management system (DMS) test-bed will serve as a proof-of-concept prototype for advanced technology and for formulation of a systems engineering methodology to deal with distributed data systems architecture. The DMS test-bed will support structured testing for critical elements and/or design concepts. Its activities will also be coordinated with, and tested against interfacing subsystems and user requirements. The DMS test-bed program will be a

multicenter effort that includes ARC (analysis and simulation), GSFC (end-to-end design), JPL (autonomy and automation), KSC (system checkout); LaRC (technology development); MSFC (system integration), NSTL (user requirements), and JSC (space station data system and overall test-bed management).

The issues addressed by these test-bed activities cover a wide range of computer technology, a field that is currently in great flux. In some areas there is substantial controversy, e.g., in system architecture, over the choice of central versus distributed processing. In other areas, e.g., software and fault diagnosis, there is a general awareness that present technology is inadequate for the scale of system size and the demands for quality. The discussions in this chapter aim to distinguish the key technology issues and to develop a framework for integrated design that can meet near-term and long-range technology requirements. We hope that the formulation of the technology issues and the design framework will provide useful information for the existing NASA programs.

E. Research and Development

There are trends in technical development that will yield significant results over the lifetime of the space station. It is vital to the future of the space station program that the IOC configuration be designed so as not to foreclose future integration of those techniques. In this section, we summarize recommendations for research and development discussed earlier in this chapter. All of these areas are subjects for near-term study (1985 to 1992), and indeed are vital to the *very near future*—before any detailed design of the SSIS is undertaken. There is also mid-term and long-term relevance in these areas, particularly with respect to ongoing evolution. (Specific recommendations are indicated in *italics*.)

In the following, we present (1) a summary of technical goals and a suggested strategy for technical development and evaluation, and (2) a set of mechanisms and techniques that can be applied to the IOC configuration to prepare it for future growth and evolution.

1. *Technical Goals*

A Distributed-Hierarchical Architecture for the SSIS. A general architectural scheme should be developed that will encourage a unified treatment of multiple SSIS requirements (performance, fault tolerance, security, safety, and the like) and provide a framework for system growth and evolution. We recommend a multilayer functional hierarchy (similar to the ISO model for communications) as a general scheme, and a generalized distributed processing organization that will allow evolution from mostly centralized to mostly distributed control. The issue of unified treatment of requirements is discussed further in Appendix C.

Software Engineering—The use of appropriate software engineering techniques will be essential to avoid runaway costs, unreliability, and inflexibility in system and application software. Critical issues for future study in this area are summarized as follows.

- *Software Development Methodology and Tools—A suitable collection of tools that enforce a comprehensive software methodology must be established. The methodology adopted should help ensure that programs will work properly (e.g., safely, securely, and reliably) under all possible circumstances. In addition, new techniques that enhance evolvability should be sought and developed. Special techniques should be developed to support onboard software production and maintenance.*

- *Programming Language Issues—A study should be made of what programming languages are potentially suitable for the SSIS, and what deficiencies would need to be overcome with any particularly desirable choice.* Special needs exist for concurrent processing and domain-specific program generation. The needs for programming distributed applications present particularly serious problems. Ada should be considered, but its known deficiencies for dealing with concurrency and multitasking, security, hardware fault tolerance, and software fault tolerance must be addressed.

Intelligent Data Management and User Assistance—An architectural approach should be developed that unifies management of all SSIS data bases and integrates data access control with other crew-machine interface functions Provision should be made for continued enhancement of user support, especially the use of artificial intelligence to automate data access and improve the quality of data representation and explanation. Important targets for near and long term development are (1) the presentation of application-oriented views of data that suppress irrelevant details of data base structure and (2) increased logical functionality in the data base system that increase its usefulness as a logically consistent model of the space station.

Intelligent Machine Agents for System Robustness. Expert system techniques should be developed for extending standard mechanisms for system robustness, such as fault diagnosis, recovery and security. These functions should be identified and given general form, e.g., a maintenance assistant, a security assistant, to encourage continuing enhancement. Provision should be made for placing them within the general hierarchical structure suggested above.

Achievement of all of the attributes described above (e.g., fault tolerance, security, privacy, integrity, timely responsiveness, and evolvability) within a unified space station computer complex represents a major problem that requires considerable awareness, advanced planning, and diligence—as well as the further study noted here. The successful simultaneous attainment of all of the required attributes represents a particularly difficult challenge.

a. IOC Design Techniques to Permit System Growth

The following sets of techniques and mechanisms to facilitate system evolution can be applied in the initial system design with relatively low overhead cost.

Software Mechanisms to Prepare for Robust Distributed Systems

Name and Authorization Manager. Provide a directory of all system objects, giving their locations and authorized users.

Intermodule Communication. Provide interface data representations and protocols to support communication among software modules.

Data Tagging. Provide packages of descriptive information associated with all intermodular data, e.g., location, time of origin, and priority, to ensure the system attributes of reliability, security, and real-time performance.

Interface Standards. Define standard interface functions for system support in communication, data access, and error handling, to encourage data sharing, system growth and relocation of modules.

Intelligent Data Management. Provide modular data management units, employ data structures that are not rigidly bound to specific hardware features, and provide support for *demon* (change detector) functions.

Software Development Techniques

Modern Software Methods. Use techniques such as modularity, hierarchy, specification, parameterization, logical synchronization, and design libraries that encourage simplicity and generality of abstractions.

Maintain a Design Knowledge Base Document the reasoning behind design decisions, and the history of the design, test, and evaluation of the system.

Graphic Input and Output. Provide logically accessible display data structures.

2. *Research Topics*

This section lists specific research topics in the technical areas distinguished above. Technology resulting from the research will help to increase the performance, reliability and intelligence of the SSIS.

Significant research questions in distributed processing that need detailed study include the following:

- How to partition processing resources so as to provide ultra-reliable support for critical space station functions.
- How to minimize interruption and loss of service and data after serious loss of processors and network continuity.
- How to program highly concurrent, dynamic (run-time determined), distributed processes to achieve objectives for throughput and real-time performance.
- How to unify the set of mechanisms needed for fault tolerance, security, data privacy and safety.

3. *Suggestions for Leading-Edge Prototype Developments*

We have identified many research issues that address critical deficiencies in several technical areas of present computer technology. In this section, we propose several prototype development efforts that would integrate significant sets of these research issues. Each effort would aim to produce a system that would serve as a focal point for the introduction of new technology over the life of the space station. Such service would be in two forms: (1) as a laboratory test bed for ongoing technical development and (2) as an operational part of the SSIS, with a scope of application that would be limited initially by cost of implementation and immaturity of technique, but would grow with advances in technology. The suggested prototypes discussed below are: (1) An intelligent, distributed data-base system, (2) The kernel of a very-high-quality distributed processing system, and (3) A software development environment for functional programming.

a. <u>An Intelligent, Distributed Data-Base System</u>

We envision a data-base system that would be a primary integrating framework for SSIS computations (astronaut information needs, operational data processing, maintenance, ground communication, and the like). It would have the following characteristics:

- An intelligent interface to the crew, capable of problem-oriented data definition and retrieval. The interface would have the

character of an "astronaut's assistant," and would provide access to a graphic IO subsystem (development of such a system would be an option in the proposed effort).

- Unified treatment of diverse data models, such as relational, network, graphic and pictorial

- Mechanisms incorporated in the data base that provide flexible execution of tests of pre-defined constraint relationships on data (also known as "alerts"); these tests are invoked whenever the relevant data are modified.

- Powerful support of fault tolerance and data recovery under minor and major fault conditions.

- Dynamic relocatability of data within a distributed system, to support rapid access to data and efficient use of network bandwidth.

b. The Kernel of a High-Quality Distributed Processing System

We recommend development of a prototype distributed processing system that would embody an extensive set of the attributes of an ideal SSIS distributed system. The system would be designed to serve as a distributed kernel of a general-purpose distributed system, i.e., it would provide a basic, highly dependable set of functions upon which system services and application services would be built. Initially, because of its lower speed, it would perform only infrequent, high-criticality service, e.g., reconfiguration and recovery. With advances in technology, speed would be likely to increase, and the scope of system responsibility would grow.

The system would have the following characteristics:

- Highly distributed control and data management.

- Support of atomic actions for a wide class of process synchronization protocols.

- Very high fault tolerance and flexible error recovery.

- Very high security.

- Enforcement of constraints on real time performance and correct sequentiality of safety-critical actions.

- Self-identifying data (tagging) to support addition of new performance attributes, e.g., acceptance testing.

c. A Software Development Environment for Functional Programming

We recommend a software development environment that supports the functional form of very-high-level programming. In functional programming, the programmer defines the results of a desired computation, with minimal specification of the exact order to be used in interpreting the definition. It enhances programming reliability and economy by supporting abstraction, and it allows maximal use of parallel processing. The environment would aim at two applications: (1) ground-based software development for SSIS system and application programs, and (2) onboard software development and generation. Features of the environment will include the following:

- A functional language for high-level, goal-oriented programming of complex applications, such as (1) operations that integrate several SSIS subsystems and (2) major fault recovery.

- Functional programming techniques to support distributed, real-time constrained, critical computations.

- Functional programming techniques for program generation in specific application domains.

- Graphic programming techniques capable of (1) representing functional programs with minimal text input and (2) providing graphic feedback to show the effects of execution with specified data.

F. Conclusions

The space station information system must meet severe requirements for performance, reliability, ease of use by the crew, and capability for evolving to meet new service demands and new technology. Meeting these goals will require solutions to a large number of problems and an architecture that allows for the integration of new techniques as additional forms of functionality.

Recommendations have been made for immediate design actions for the IOC configuration and for longer-range research studies. Areas of highest priority are

(1) Distributed Processing—Integrated design for multiple attributes, recovery from massive fault conditions, high-order concurrency.

(2) Intelligent Data Management—Uniform representation of heterogeneous data, intelligent data definition, retrieval and explanation, logical support for data consistency.

(3) Software Development—Very-high-level design languages, design libraries, application-oriented software generation.

(4) Fault Testing and Diagnosis—Efficient testing over all system levels, correct diagnosis of unusual fault conditions.

Of these, Items 1 and 2 would yield the greatest return in future computing power and user convenience. Item 3 is indeed crucial, but research being done elsewhere is pursuing similar objectives. NASA should, therefore, monitor progress in software methodology to ascertain the merits of a major investment in advanced software techniques. While there is current research in advanced fault diagnosis techniques, the possibility of unusual fault modes and of a lack of expert maintenance in the space station may justify NASA investment in knowledge-based techniques for fault diagnosis.

Our recommendations for the IOC configuration have emphasized measures that would provide immediately useful properties and also ensure architectural frameworks that would allow enhanced capabilities and attributes. These include: (1) software mechanisms that allow gradual transition toward systems with more distributed processing and control; (2) support for hierarchical structure with well-defined interface functions for major services; and (3) support for properties such as fault tolerance and security that cannot be imposed subsequently on an existing system.

IX. SPACE STATION INFORMATION SYSTEM

REFERENCES

1. TRW Space and Technology Group, Redondo Beach, California, *"Space Station Data System (SSDS) Third Quarterly Review"*. Vue foils, TRW presentation to NASA at NASA JSC, 12 February, 1985.

2. McDonnell-Douglas Aircraft Corp., *"Space Station Data System (SSDS), Third Quarterly Report"*. Vue foils, MDAC presentation to NASA at NASA JSC, 13 February, 1985.

3. "Advancing Automation and Robotics Technology for a Manned Space Station and the U.S. Economy," (Preliminary Strawman Draft), Washington, D.C., NASA Headquarters, revision of 14 December 1984. NASA Advanced Technology Advisory Committee (ATAC).

4. G. Le Lann, "On Real-Time Distributed Computing," *Information Processing 83*, pp. 741-753, IFIP (1983).

5. N.G. Leveson, "Software Safety in Computer Controlled Systems," *IEEE Computer*, February 1984, *17*(2), pp. 48-55.

6. P. Bernstein, N. Goodman, "Concurrency Control in Distributed Database Systems," *Computing Surveys*, June 1981, Vol. *13*(2).

7. D. L. Parnas, "On a 'Buzzword': Hierarchical Structure," *Information Processing 74 (Proc. IFIP Congress 1974)*, pp. Software: 336-339 (1974).

8. K.N. Levitt, L. Robinson, and B. Silverberg, "The HDM Handbook, vols. I,II,III," (Tech. Rep.), SRI International, Computer Science Lab, 1979.

X. MAN-MACHINE INTERFACE

X MAN-MACHINE INTERFACE

A. Introduction

The operation of the space station will depend on many systems that must be monitored and controlled by people, both onboard and on the ground. A range of interaction methods must be made available for this task, [1, 2]. Graphical displays and animation are required for effective information presentation, [3, 4, 5]. Routine data-base access could take the form of selection of stored reports. Data-base inquiry, the kind often needed in trouble-shooting, requires the ability to find answers that satisfy complex descriptions, [6]. Although users could be taught formal query languages, data-base access would be made much easier through the use of natural language, both spoken and typed, [7, 8]. The greater abstraction of natural language over formal query languages allows much of the detail of the organization of the data base to be hidden from the user. Natural language technology could be used in many applications on the space station. Complex systems, such as onboard manufacturing operations, could be controlled by a combination of natural language queries and commands associated with graphical displays. Users would be most comfortable using spoken language. Language understanding systems that are attuned to a particular speaker (speaker-dependent) would probably be adequate for most tasks.

The man-machine, sometimes called the operator-system interface (OSI), facilitates communication between the space station crew and the hardware/software system that monitors and controls. In this chapter, we describe the devices and techniques used in the dialogue between man and machine and identify the human-factors problems that must be solved to achieve an effective interaction.

B. State of the Art

1. *Human-Factors Considerations*

The traditional approach to human engineering, or ergonomics, has in the past emphasized "knobs and dials"—fundamental issues of legibility and operability as determined by the basic operating characteristics of the human being. More recently, the advent of computer technology has permitted the system designer to automate some tasks previously performed by the human, thus eliminating even the potential for human error in the performance of those tasks, and often enhancing overall system performance. However, in some cases the application of automation technology not only has resulted in the creation of

entirely new opportunities for human operator error, but has reduced the performance of the complete system. Some have argued that experience to date with large, complex, semi-automated systems suggests that we may have reached some limits in our ability to integrate effectively complex technology and the human operator, and that we must develop better understanding and new approaches to human-machine system design.

A recent report of the Aeronautics and Space Engineering Board [9] discusses machine-aided information acquisition, processing, and decision-making:

> The panel believes that a new discipline is emerging that will have a profound impact on how we will design future systems ... The goal is to develop the understanding required to establish principles and practices for designing systems that make "best use" of the human, are tolerant of human error, and reduce the costs of systems and training ...
>
> The trend in the application of technology in present-generation aircraft is to relegate closed-loop control tasks at all levels to microprocessors and automated actuators, and to limit the role of the human operator to that of monitor and manager whose primary function is to intervene in the event of system failure. Available data about human performance in such systems suggests that this approach to complex human-machine system design will not be optimal. If applied fully, this design approach forces the human operator into a monitoring role for which he is not well-suited. It fails to make effective use of the human's ability to work with uncertain, incomplete, and ambiguous date—a task for which machines are not and may not become well-suited. At present it is unclear whether future systems would be designed to augment the human operator's information acquisition and processing capabilities, or whether they would be designed to eliminate the human from the system, except to provide backup in the event of system failures. Although considerable effort has been expended in studying this question, it has been mostly of an *ad hoc* nature; more focused or organized effort is needed to clarify the issue.

2. *Devices and Techniques*

In this section we describe some of the newer OSI elements such as natural and spoken language communication, displays and pointing devices. The interaction between man and machine is divided into input devices, the interface module, and output devices.

a. Input Devices

Traditional input devices include keyboards (for text input), microphones (for audio input), cameras (for video input), and scanners (for graphic input). The technology used is stable and well known. With the advent of interactive computing, the need arose for pointing devices such as the joystick, the mouse, and the touch-screen. All of these devices require finger contact with a terminal. However, in situations such as module repair, it is often inconvenient to have physical access to a terminal, and remote pointing devices may be necessary. Two remote pointing devices—eye tracking and finger pointing devices—are now possible and are surveyed below.

Eye-Tracking Devices Two major eye-tracking devices exist: fixed and head-mounted. Fixed eye-tracking devices are very accurate, (about one minute of resolution) and are currently used mainly for medical purposes. Head-mounted devices are probably more useful for man-machine interface purposes but typically achieve only a one-to-two-degree accuracy. Eye tracking is complex because of the many degrees of freedom involved—those of the head itself and those of the eye in relation to the head. A typical head-mounted device, therefore, contains one monitoring mechanism for each eye (relative to the head) and a single mechanism monitoring the position of the head.

Head tracking can be done in two ways. A small transmitter can be attached to the head, with receivers sensitive to direction located in the surrounding room. A second way is to place both transmitters and receivers in fixed positions in the surrounding area and mount a reflecting device (typically with magnetic properties) on the head.

The corneal-reflection eye-tracking technique, used in head-mounted eye trackers, is based on a virtual image produced by the cornea and on the fact that the radius of curvature of the cornea is less than that of the eye. Typical implementations make use of a light source (often unseen light, such as infrared) mounted in a fixed position relative to the head and an optical system to track the virtual image. Knowledge of the location of this virtual image and the light source uniquely determines eye position. Early implementations of eye trackers were cumbersome and inaccurate owing to the instability caused by the weight of the device. Recent ones, using fiber optics and microprocessors, separate the sensing part from the rest of the system, resulting in a much lighter head mount. The use of head-mounted eye tracking would be useful in many space station applications. In particular, when a helmet must be worn, the addition of an eye tracker is simple, not only because of the mounting procedure but because head position tracking is easier and because the field of view is limited.

The greatest limitation of head-mounted eye trackers is their accuracy,

which is about one or two degrees and is not likely to improve substantially in the near future. With such limited accuracy, items pointed at must subtend at least four degrees. When a terminal screen is viewed from 30 in. objects are required to be 2 in. by 2 in., which is rather large for a standard 12-in. display. To overcome this problem, larger screens or "zooming-in" procedures are used. With little operator training almost all items can be zoomed in on and located within one or two steps. However, when physical contact with the terminal is possible, standard pointing devices, such as a mouse, outperform an eye tracker in accuracy of object location.

Remote Finger Pointing. Finger pointing, despite its low resolution, is extremely efficient. It is advantageous from an operator-machine interface standpoint because it does not require operating a device such as a mouse or joystick. Touch-screens are an example of finger pointing devices but, as mentioned before, require physical proximity to the screen. Recent research and development has produced a new approach—remote finger pointing (RFP), consisting of a small device attached to the user's finger. The device transmits a narrow beam, and the receiver consists of a plane sensitive to the transmitted waves. Several problems that still plague the designers of RFP devices are:

- Producing a narrow enough beam to achieve sufficiently high pointing resolution at various distances.
- Designing a transmitter that does not burden the operator (in terms of weight and shape).
- Integrating the receiver plane with a display device (e.g., using a transparent receiver plane with a standard display screen).

RFP devices are more promising than eye trackers as pointing devices because it is much easier for a human to control the motion and stability of his fingers. (It is actually impossible to have the eye stand still). In addition, an RFP device is physically more comfortable for the operator. No technological breakthrough is required to bring about an RFP device; it is mainly a problem of development and integration of new components using existing technology.

b. Interface

We have divided the interface into an input and an output direction.

Input Direction.

Natural Language Understanding—Natural language understanding is an important issue in the operator-machine interface. The words "natural language" imply that users would use a language identical with, or at least very

similar to, spoken language and not a traditional command language. The word "understanding" implies that the system can extract the user's intent from the message.

Several natural language understanding systems are available today and all work reasonably within a limited task domain and a well-defined context. For example, natural language can be used to query a data base; the interface translates natural language sentences into the formal query language and displays the answer. Currently, most natural language understanding systems interface with a single data base. Attempts have been made to construct a single interface to several data bases with the interface itself deducing which data base to interrogate, and such devices will soon be available.

In spite of limitations, natural language interfaces to data bases are extremely useful for mission-oriented activities when confined to a specific activity. Many expert system applications fall into this category—repair and maintenance of known subsystems, navigation, power management, and more.

Performance of natural language understanding systems will increase substantially when new hardware architectures are introduced. At the execution level, natural language understanding requires massive searches and comparisons that can be carried out in parallel to reduce execution time. The new computer architectures being funded by the DARPA program will be well-suited for natural language understanding purposes. Specially-designed multiprocessors dedicated to natural language understanding will some day be part of every computing system.

Speech Recognition—Because of the important role that speech plays in human communication, speech recognition, also called ***speech understanding***, has long been the subject of research and development. Three major issues concern speech recognition researchers: speaker independence, vocabulary size, and isolated- versus connected-speech processing. Speaker independence is the degree to which the system can recognize speech spoken by arbitrary speakers. While the goal is to achieve complete speaker-independence, this has not been attained by most available systems. Current speaker-independent systems support only a very limited vocabulary.

Vocabulary size impinges on both storage requirements and processing time. As the vocabulary increases, the fast storage requirements also increase. To increase recognition accuracy, more storage is needed for each item in the vocabulary. Larger vocabularies also require a longer processing time, thereby limiting usefulness in an interactive environment.

Isolated-word recognition systems have been far more successful than

continuous speech recognition. In isolated-word recognition systems, each word must be clearly delimited (e.g., by a pause); each word is then treated as an independent entity and processed separately. Continuous speech requires that word boundaries be identified automatically—a complex task that has eluded solution. Currently, systems are available that operate in real time recognizing a vocabulary of about 1000 words of isolated speech. These systems are speaker-dependent, and the amount of training varies among the systems. Similar systems for recognizing 5000 words are possible, but not yet available. One of the major problems in such large systems is the long and tedious training sessions required. It is not clear that there are isolated word-recognition applications requiring a large vocabulary.

Speaker-independent systems exist, but their vocabulary is limited to few tens of words. Constructing a speaker-independent system to recognize the spoken alphabet is not yet sufficiently reliable. The alternative of having the operator spell words not in the vocabulary is not a viable option in the speaker-independent case, but is feasible in a speaker-dependent system with proper isolation.

Because only about 150 matching-templates are required for spoken-digit recognition, continuous spoken digit-recognition systems have been developed and are available. These systems are useful for inventory control, service and repair, and the like. Interestingly enough, most existing systems are quite insensitive to surrounding noise as long as a single dominant speaker can be identified. This means that speech recognition can be achieved in the presence of mechanical noise such as exists in a machine room.

Part of the DARPA SCP is a new program in speech recognition, as shown in Table 6. The program goals are to achieve accurate, continuous speech recognition for large vocabularies (about 10,000 words). Significant progress in this area can be expected in the next decade.

Output Direction

Speech Synthesis—In speech synthesis, a computer generates audible words for output instead of printing or displaying the words. Advantages to speech synthesis over printed output are naturalness of communication and the ability to communicate by listening, allowing the person involved to use vision for other tasks.

Two fundamentally different types of speech synthesis systems may be identified: limited vocabulary and unlimited vocabulary. Limited-vocabulary systems use prerecorded, digitally-encoded speech, often organized as individual

Table 6: DARPA Strategic Computing Plan for Speech and Natural Language

Functional Objectives for Speech Subsystems

FY 86	Recognition of words from a 100 word vocabulary for a given speaker under severe noise and moderate stress conditions
FY 88	Recognition of sentences from a 1000 word vocabulary with moderate grammatical constraints in a speaker adaptive mode under low noise and stress conditions
FY 89	Recognition of connected speech, independent of speakers, from a 200-word vocabulary with strict grammatical constraints under severe-noise and high-stress conditions.
FY 92	Recognition of sentences, independent of speakers, from a 10,000 word vocabulary with natural grammar under moderate-noise and low-stress conditions

Functional Objectives for Natural Language Subsystems

FY 86	Natural language interfaces with some understanding of commands, data inputs, and queries (e.g., interface to a data base and to an expert system)
FY 88	Domain-specific text understanding (e.g., understand paragraph-length intelligence material relating to air threat)
FY 90	Interactive planning assistant that carries on task-oriented conversation with the user
FY 93	Interactive, multiuser acquisition, analysis, and explanation system that provides planning support and substantive understanding of streams of textual information

words. Unlimited vocabulary systems take arbitrary text as input, synthesize an acoustic waveform via appropriate algorithms, and produces a sound output. The algorithms analyze the input text and transform it into one or more intermediate representations, such as a phonetic representation, based on a set of rules. The actual acoustic waveform is derived from an intermediate representation. The major limitation of such a system is that any set of rules will cause mispronunciation when dealing with a large, inconsistent, natural language like English To circumvent this problem, most of these systems use an exception dictionary that contains the pronunciation of words that are exceptions to the internal rules.

Limited-vocabulary systems have been attractive in the past for applications in which relatively small vocabulary size is not a drawback and high speech quality is required. Unlimited-vocabulary systems have recently begun to approach human speech in terms of comprehension, if not overall quality, and will be appropriate for the space station.

Video Synthesis—Pictorial information on the space station will originate electronically from two sources: video and computer graphics. In the following, we discuss these two sources and their combination.

Videotape recorders have become widely available at low cost. However, the most promising video technology for the space station, the videodisk, is less well known (see Appendix B). Videodisk can produce either full motion video or equally high quality still video. In contrast to videotape, videodisk is a high speed random-access device that may be easily computer-controlled, and has an extremely high information density. These qualities, combined with computer control as discussed below, make it a good candidate for storing and presenting information relevant to the space station.

Computer graphics has been rapidly developing for the past several decades. There are now systems available that will produce complex, real-time motion, three-dimensional graphic images with standard television resolution. The cost of such systems is within the budget of a small laboratory. General trends in more cost-effective hardware have been helpful in producing this capability with custom VLSI chips playing an essential role. The graphics-standards efforts (notably the ACM CORE and the ANSI GKS) will also be important in promoting increased use of graphics

For the space station, the combination of the videodisk and computer graphics holds the most promise. To illustrate this, let us imagine a repair task aboard the space station. Support hardware consists of a computer graphics system, a videodisk, and a display, all connected to a computer. Images of

components to be repaired are stored on the disk, including sequences of manipulations recorded during a simulated repair. The computer has information about the location of items on the video-disk and in individual video-disk images. It is able to select an image or sequence on the disk, present it on the display, and superimpose computer graphics information on the video image. This graphics information could be as simple as a showing a specific component, or as complex as providing a running commentary on repair of the component, along with graphics showing exploded or internal views. Thus, a particular repair sequence can be viewed at any of several levels of detail, or specific assistance can be obtained for a particular aspect of the repair.

C. Space Station Applications

The space station is an example of *supervisory control* in which the crew interacts via a computer with a complex and semiautomatic process, setting initial conditions for, intermittently adjusting, and receiving information from a computer that closes a control loop through external sensors, effectors, and the task environment. There are two main topics that arise in the design of such *man-machine interaction* subsystems: (1) the technology of the input/output devices, and (2) the human factors problems that arise in making effective use of these devices. The first of these involves equipment such as displays, keyboards, light pens, joysticks, graphical input tablets, printers, and speech input/output devices. The latter is concerned with improving collaboration between the human and the computer.

Some of the space station activities requiring man-machine interactions are real-time command and control; passive and active monitoring; information storage and retrieval; computational support; process planning and scheduling; recovering from failure; control of experiments and of manufacturing processes; and communication. The conventional equipment that must be provided for interaction includes the spectrum of displays and input/output devices indicated above. In our discussion, we will stress the less conventional use of natural language, both spoken and typed. Many applications can be found for natural-language technology on the space station:

- in EVA when the use of keyboard is impractical
- in repair tasks when the hands of the astronaut are occupied
- in control of complex systems, such as onboard manufacturing operations
- in information retrieval, where natural language input/output avoids the need to learn special formal query languages

Human factors problems arise in the space station because as automation is introduced, the following situations may arise:

- the astronaut must retain a high level of alertness while controlling a system that, under normal operating conditions, requires only occasional fine-tuning of system parameters to maintain satisfactory performance

- the astronaut must be able to serve as a backup when critical failure or malfunction occurs, yet such important participation of the astronaut occurs only infrequently and at unpredictable times

- in dangerous situations, the time constraints associated with participation can be very short, of the order of a few seconds or minutes, and there is little time for an astronaut to get "up to speed" concerning the situation

- good performance requires rapid assimilation of large quantities of information and the exercise of relatively complex inference procedures

D. Demonstrations

Demonstrations of man-machine interactions concerned with teleoperation were discussed in Chapter 5; here we focus on natural language demonstrations. The following demonstrations would indicate the existence of the needed capabilities:

- Near-Term (1985 to 1990)

  - Demonstrate natural language access to databases, with speaker-dependent voice input over a vocabulary of 1000 words.

  - Demonstrate natural language control of a complex system, such as a factory.

  - Demonstrate simple acquisition facility for an expert system in natural language.

  - Demonstrate useful recovery facilities for handling common grammatical errors.

Medium-Term (1990 to 1995)

- Demonstrate natural language control of a complex system, including the ability to engage in extended dialogue.

- Demonstrate speaker-independent voice input over vocabulary of 1000 words.

- Demonstrate speaker-dependent voice input over vocabulary of 10,000 words.

E. Research and Development

1. *Man-Machine Devices R&D*

Research results in speech and natural language will be available from sources such as the DARPA SCP program to satisfy the needs of the above demonstrations. Examples are connected speaker-dependent speech recognition; syntactic analysis and parsing with speech input; semantic representation of sentences; and models of dialogue, including recognizing and reasoning about the knowledge and plans of the system's user. NASA research and development efforts in language and speech should extend the above efforts by concentrating on the special problems found in the space environment. Some of these developments may be appropriate for the Space Station Program Office.

2. *Human Factors R&D*

The Committee on Human Factors, sponsored by the Office of Naval Research, the Air Force Office of Scientific Research, the Army Research Institute, and NASA, identifies basic research needs in support of human-factors-engineering applications and makes recommendations for basic research. Their 1983 report [10] states:

The human factors aspects of supervisory control have been neglected. Without further research they may well become the bottleneck and the most vulnerable or most sensitive aspect of these systems.

Some of the research results required to avoid this vulnerability are:

- How to display integrated dynamic system relationships in a way that is understandable and accessible.

- How to provide the operator with a natural means of indicating to the system what is desired and why.

- How to aid the operator's cognitive process by providing

computer-based knowledge structures.

- How to coordinate several operators in control of the same system.

- How to improve the man-machine dialogue in a large, complex, interactive system, as experience is gained in the use of the system.

Close collaboration between human factors researchers and the designers of the space station will be required to address these problems.

REFERENCES

1. J. Martin, *Design of Man-Computer Dialogues,* Prentice-Hall, 1973.

2. B. Shakel, *Man-ComputerInteraction: Human Factors Aspects of,* Sitjthoff and Noordhof, The Netherlands, 1981.

3. S K. Card, W.K. English, and B.J. Burr, "Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT," *Ergonomics*, 1978, *21*, pp. 396-410.

4. J. Ketchel, "Visual Display Terminal Research--the Opportunity and the Challenge," *Human Factors Society Bulletin*, 1981, *24*(10), pp. 2-3.

5. H. Ledgard, A.Singer, and J. Whiteside, *Directions in Human Factors for Interactive Systems,* Springer-Verlag, New York, 1981.

6. P. Reisner, "Human Factors Studies of Database Query Languages. A Survey and Assessment," *Computer Surveys*, 1981, *13*, pp. 13-32.

7. V.A. Riley and R.E. Vestewig, "A Voice Interactive System for Aiding and Documentation of Space-Based Tasks," *Computers in Aerospace Conference*, pp. 171-177, American Institute of Aeronautics and Astronautics, New York, New York, Hartford, Connecticut (1983).

8. V.A. Riley, D.R. Baum, and R.E. Vestewig, "Voice Interactive Maintenance Aiding Device: Executive Summary Report," (Tech. Rep.), Minneapolis, Minnesota, Honeywell Systems and Research Center, October 1983. Contract MDA 903-81-C-0173.

9. "Aeronautics Technology, Possibilities for 2000," (Tech. Rep.), Washington, D.C., National Academy of Sciences, 1984. Report of a Workshop.

10. "Research Needs for Human Factors," (Tech. Rep.), Washington, D.C., National Academy of Sciences, 1984. Prepared by the Committee on Human Factors.

Appendix A

Pacesetter Technologies for Telepresence and Robotics

A. Pacesetter Technologies for Telepresence and Robotics

Appendix A

Pacesetter Technologies for Telepresence and Robotics

This appendix lists specific examples of equipment and techniques related to the six categories of pacesetter technologies discussed in Section V-C, page 68.

Equipment

Input Equipment

- Smaller and lighter transducers of all kinds (e.g., fingertip-sized color-television cameras, fiber-optic pressure sensors)
- Faster three-dimensional imaging methods (probably based on lasers, VHSIC technology, and electro-optical signal processing).
- Phased-array methods to scan ultrasonic beams for acoustic range sensors.
- More rugged tactile arrays, both in fingertip size and as large, flexible sheets.
- Distributed proximity sensors for detecting the nearness of arbitrary objects (e.g., to prevent collisions) [1].
- Proprioceptors to measure the deformation of limber structures under load (perhaps distributed throughout a structure)
- Improved speech-processing devices (e.g., speaker-independent and capable of recognizing natural, continuous speech rather than just isolated phrases).
- Data communication devices based on modulated infrared light, ultrasound, or other forms of energy suitable for the space station environment.
- Methods for reducing the number of wires carrying sensor data.
- Standards for mountings, connections, and data transmission.

Output Equipment

- Smaller and lighter transducers of all kinds (e.g., actuators made from piezoelectric or phase change materials [2])
- Actuators with a higher strength-to-weight ratio (e.g., chemically powered but electrically controlled, with the performance of a hydraulic actuator but without pumps or hoses).
- Actuators that do not require brakes, power transmission devices such as belts, or speed reduction mechanisms such as gears.
- Actuators with built-in sensors for position, speed, acceleration, effort, temperature, and other parameters.
- Actuators for control of limber structures (perhaps distributed throughout a structure).
- Lighter, stronger, more rigid structural members for articulated mechanisms, such as arms and legs.
- More dexterous hands for teleoperators and robots.
- "Feet" suitable for legged locomotion on and in the space station and other large orbital structures.
- End-effector designs incorporating generalized versions of the remote-center compliance (RCC) concept.
- Fail-safe or even self-healing actuators.
- Improved display hardware, such as the following:
    - Capable of showing moving, colored, shaded solid objects in real time.
    - Smaller, thinner, lighter, and low-powered displays.
    - Head-mounted and head-up displays
    - Displays that can be built into a space suit helmet—e.g., in the visor.

- Speech synthesis equipment that produces more understandable speech.

- Data communication devices based on modulated infrared, light, ultrasound, or other forms of energy suitable for the space station environment.

- Methods for reducing the number of wires carrying control signals.

- Standards for mountings, connections, and data transmission.

Control

Interpretation of Sensor Data

- Use of CAD data bases to provide object shape information to aid identification and pose measurement.

- Faster, more reliable, more versatile model-based methods of matching sensor information with geometric models for object detection, identification, location, and inspection [3].

- Improved methods for building geometric models from sensor information.

- VLSI chips for complex signal processing in the sensor ("smart sensors") to reduce requirements for communication bandwidth and data storage.

- Methods for measuring the inertial parameters of an object held in the gripper of a manipulator for improved control.

- Theory for multisensory integration—especially vision and tactile sensing—for object identification, location, and inspection.

- Interpretation of tactile images to ascertain the orientation of an object in the gripper.

- Interpretation of forces at the fingertips to determine and control grasp.

- Methods for detecting jamming and wedging when putting parts together.

- Use of visual and tactile textures in recognition, location, and inspection.
- Visual navigation methods.
- Methods for reading gauges and other human-readable indicators.
- Theories of interpretation (e.g., a concise and unambiguous formalism for describing multisensory inspection or pose measurement procedures so they can be generated, analyzed, compared, and optimized).

Adaptive Control

- Use of geometric information from CAD systems to help in deciding what actions to perform.
- Faster, more accurate visually guided manipulator control for grappling, docking, part mating and fitting.
- Integration of visual and tactile sensing in multifingered hand control.
- Automatic sensor-guided tracking of moving objects and physical contours.
- Simulation of different kinds of mechanical compliance at the gripper by sensory control of motions (e.g., to turn cranks, operate latches, and damp down the spin of a satellite).
- Model-based (i.e., *open-loop*) collision avoidance methods for mechanism control [4].
- Sensor-based (i.e., *closed-loop*) collision-avoidance methods for mechanism control.
- Dynamic collision avoidance algorithms that take account of the mass and inertia of an object held in a gripper, the strength of joint actuators, flexibility of the arm segments, etc. This would be particularly useful in berthing large payloads.
- Coordination of multiple articulated mechanisms (e.g., for collision avoidance and cooperative handling) [5]

- Reliable methods for handling limp materials.
- Reliable methods for handling low-friction ("slippery") materials.
- Theory of contact phenomena (e.g., bouncing) for automatic docking and mating of large structures.
- Automatic calibration methods for visual and tactile sensors.
- Theory of error analysis in precise sensor-guided manipulation [6].
- Theory of sensor-guided deformation processes in assembly and manufacturing (i.e., how to bend parts into desired shapes).
- Integration of visual and tactile feedback in fitting and fastening operations.
- Measurement of physical texture by moving tactile sensors.
- Visual and tactile sensing in legged locomotion on and within complex structures.
- Sensor-guided locomotion (i.e., navigation)
- Theory for legged locomotion in zero gravity on large structures (e.g., foothold selection, route planning, how to combine with free flight).
- Simpler methods of programming adaptive-control procedures (e.g., simplified programming languages and "user-friendly" programming environments).

Generation of Output Device Control Signals

- Use of geometric data in deciding how to operate a mechanism (manipulator, leg, etc.)
- Kinematic computations based on mechanism geometry (e.g., computing the joint positions for a given hand position and vice versa).
- Theory of operation of highly-redundant mechanisms (e.g., a *tentacular* arm with many joints, or a multibranched, treelike arm mechanism).

- Dynamic stability theory for rigid mechanisms (e.g., conventional manipulators) [7]

- Self-optimizing arm joint servo control methods that adapt to the inertia of the load being manipulated.

- Theory of control of limber mechanisms (e.g. long, thin manipulators like an orbiter's RMS).

- Attitude control and stabilization in free flight, using the motion of articulated appendages alone.

- Generation of ballistic trajectories by using manipulators and legged propulsion systems (i.e., "throwing" and "jumping")

- Theory of multifingered hand control for dexterous manipulation (e.g., how to rotate an object held in the fingertips, or exert a prescribed force vector on it).

- Control theory for cable-based locomotion (e.g., deployment, use, and stowage of grappling hooks and artificial "spiderwebs")

- Control theory for zero-gravity flight in pressurized areas (e.g., using ducted-fan thrusters).

Reasoning

Reasoning encompasses the various categories of artificial intelligence: image understanding, natural language, expert systems, and automatic planning. It will be used mostly in robotics, but possibly in supervisory control to some extent as well. It makes use of such processes as logical deduction, probabilistic inference, search among alternatives, hypothesis formation, temporal and spatial reasoning [8], pattern matching, and learning.

Pacesetter technologies in this area include the following:

- A comprehensive theory of spatial reasoning, together with a compact representation for geometric information and efficient computational algorithms for use with it [8].

- Similar methods for reasoning about temporal relationships.

- Commonsense methods for qualitative reasoning about perception, manipulation, and locomotion without extensive quantitative

calculations.

- Expert systems of many kinds for specialized kinds of perception, generation, control, and man-machine interface management.

- Practical and efficient methods for automatic generation of plans for complex procedures that may involve parallel and conditional action sequences, temporal constraints, and shared resources.

- A comprehensive theory of manipulative processes to support such planning for handling, service, construction, inspection, and repair tasks.

- Methods for monitoring execution of plans, detecting problems and failures, and identifying their causes.

- Methods for rapidly modifying a complex plan to forestall problems or in the event of failure, to ensure successful completion.

- Methods for generalized machine learning based on data base access, on both formal and casual instruction by people, and on experience.

Many of these technologies would also be useful in other space station applications, such as in planning the functions and activities of the crew.

Man-machine Interface

Among *man-machine interface* technologies we include teleoperator controls, the remote-sensing methods of telepresence, and methods for specifying activities to be carried out automatically. Natural language, a category of artificial intelligence, will have its major impact in this area of space station automation.

Some of the pacesetter technologies relevant to the man-machine interface on the space station are the following:

- Methods for using geometric information from the CAD data bases to aid teleoperation, telepresence, and robot control.

- Making use of computers in teleoperation and telepresence systems to reduce the workload on the operator (e.g., to perform routine or repetitive activities automatically).

- Methods for warning the operator or overriding his commands if they are unsafe (e.g., for automatic collision prevention in remote manipulator control).

- More dexterous grippers on slave arms for operators to control.

- Improved teleoperator master controls (smaller, lighter, less obtrusive, with force feedback, higher resolution, multifingered gripper controls, tactile pattern feedback [9] to the operator's fingertips).

- "High-fidelity" telepresence (e.g., a low-fatigue, high-resolution, three-dimensional television display, or high-sensitivity force feedback).

- Methods for controlling autonomous robotic systems that do not require skill in computer programming, mathematics, or "spatial relationships."

- Improved computer-graphic display methods for telepresence, predictive displays to overcome time delays, simulation of robotic activity, and specification of inspection procedures.

- Methods for easily combining information about motion or forces generated with kinesthetic controls into a program written in a robot control language.

- Intuitive methods by which nonprogrammers can specify a procedure, which may require tactile or visual guidance, for a dexterous hand to execute.

REFERENCES

1. C. Wampler, "Multiprocessor Control of a Telemanipulator with Optical Proximity Sensors," *The International Journal of Robotics Research*, Spring 1984, *3*(1), pp. 40-50.

2. C.F. Ruoff, "Memory-Metal Electromechanical Actuators," (Technical Support Package for JPL Invention Report NPO-15960/5414), Pasadena, California, Jet Propulsion Laboratories, Winter 1983. NASA Contract No. NAS 7-100. Also in *NASA Tech Briefs* No. 2, Vol. 8, Winter, 1983, item No. 62.

3. R.A. Brooks, "Model-Based Three-Dimensional Interpretations of Two-Dimensional Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, March 1983, *PAMI-5*(2), pp. 140-150.

4. R.A. Brooks, "Planning Collision-Free Motions for Pick-and-Place Operations," *The International Journal of Robotics Research*, Winter 1983, *2*(4), pp. 19-44.

5. W.T. Park, "State-Space Representations for Coordination of Multiple Manipulators," *Proceedings of the 14 th International Symposium on Industrial Robots*, pp. 397-405, Göthenburg, Sweden (2-4 October 1984).

6. R.A Brooks, "Symbolic Error Analysis and Robot Planning," *The International Journal of Robotics Research*, Winter 1982, *1*(4), pp. 29-68.

7. W.J. Book, "Recursive Lagrangian Dynamics of Flexible Manipulator Arms," *The International Journal of Robotics Research*, Fall 1984, *3*(3), pp. 87-101.

8. "Representation and Processing of Spatial Knowledge," (Computer Science Technical Report TR-1275), College Park, Maryland, Computer Science Department, University of Maryland, May 1983. Sponsored by NASA Grant NAG5-290.

9. "A Reading Eye for the Blind," *Optical Spectra*, February 1971, pp. 18-22.

Appendix B
TELEOPERATION AND ROBOTICS SCENARIOS

B. TELEOPERATION AND ROBOTICS SCENARIOS

Appendix B

TELEOPERATION AND ROBOTICS SCENARIOS

1. Introduction

In this appendix we use an antenna deployment task as the basis for scenarios that describe how the task would be accomplished using telepresence, adaptive robotics, and an autonomous robot. These scenarios provide insight into functional requirements for each level of automation. In Section B-2, we describe the presently-planned EVA procedures for parking and then redeploying the high-gain antenna (HGA) boom on the Gamma Ray Observatory (GRO) satellite, and what can go wrong in the process. In Section B-3, we discuss the use of telepresence technology to automate this task. In Section B-4, we do the same for adaptive robotics, and in Section B-5 for artificial intelligence. Each of these last three sections contains a scenario that is intended to be illustrative rather than definitive.

2. Problem Statement

According to current plans, before it can be refueled the GRO must first be docked onto a turntable on a berthing platform on a cradle in the shuttle bay. However, the satellite has a high gain antenna (HGA) on a long boom. Because this antenna would collide with the keel of the shuttle in its normal deployed position, the refueling scenario includes an EVA to fold it up into a parking position away from the keel before docking. An astronaut would go out and undo several screws, manipulate the boom into the parking position, and then lock it there by putting a captive pin through a clevis. Figure B-1 shows the initial position of the boom when the satellite is launched. After insertion into orbit, a small stepper motor turns the driving link of a two-bar toggle mechanism to swing the boom counter-clockwise about 135° around the removable hinge pins to its normal deployed position (Figure B-2). Because, in this position, the toggle mechanism is on-center, it can no longer be back-driven by pushing on the boom, thus locking the boom in the deployed position.

Early in the refueling procedure, the RMS grapples and stabilizes the GRO. An astronaut in EVA then unscrews two bolts on the keel trunnion to release the bracket that carries the toggle linkage and the boom. Then he rotates the entire assembly as a unit about 45° counter-clockwise around the two removable hinge pins. This brings the boom to the parking position as shown in Figure B-3. Finally, the astronaut inserts a captive pin through a clevis on the boom and

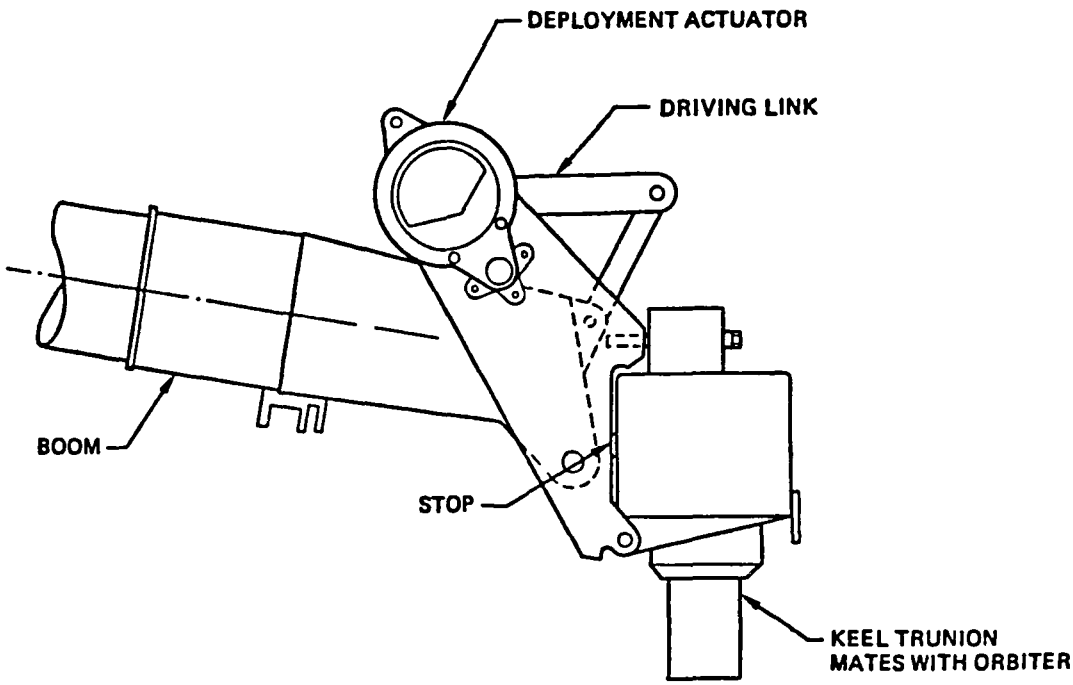


Figure 1: High Gain Antenna Boom (Pre-Launch Stowed Position)

through a hole plate on the keel trunnion to retain the boom in that position. The RMS then docks the GRO with the turntable in the shuttle bay.

After the satellite has been serviced and refueled, the RMS pulls the satellite off the turntable and moves it out of the bay. While it holds the satellite in this position, an astronaut goes out again and, reversing the above procedure, puts the boom back in the deployed position of Figure B-2.

The scenario for boom parking is the same for all three automation systems, namely:

(1) *Dock*—Bring the system into contact with the GRO.

(2) *Rigidize*—Attach it rigidly to the GRO.

(3) *Park*—Park the boom.

(4) *Unrigidize*—Release the rigid attachment.

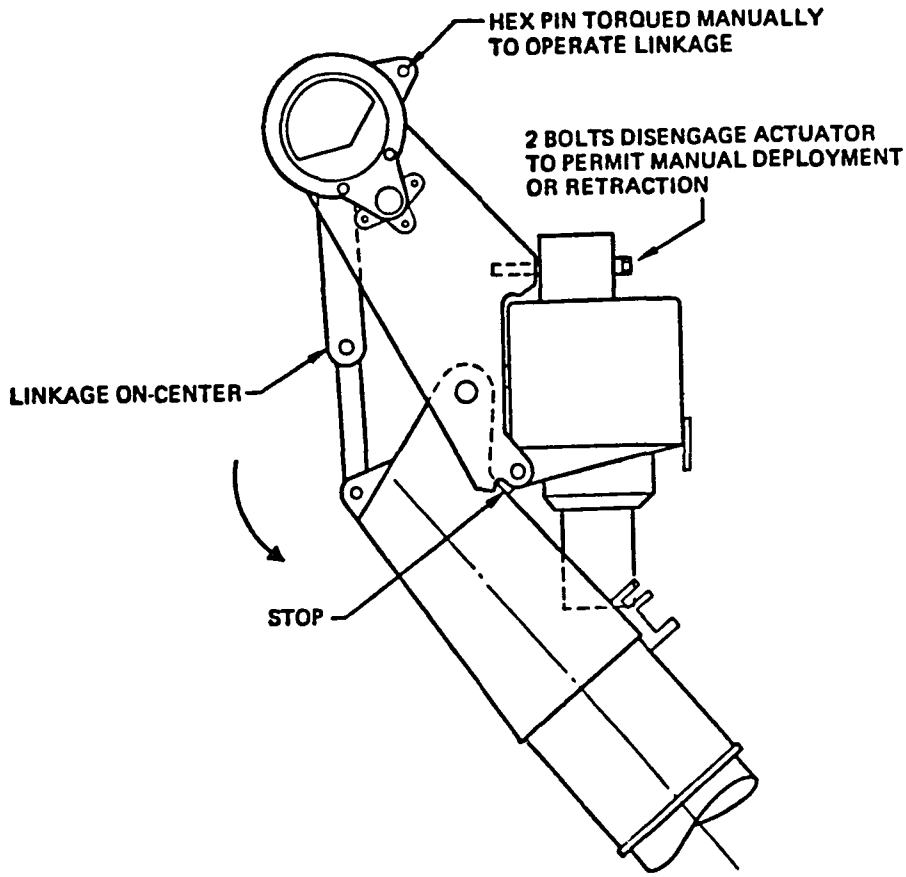


Figure 2: High Gain Antenna Boom (In-flight Deployed Position)

(5) *Undock*—Move the system away from the GRO.

Parking the boom is itself a three-step process:

(1) Loosen the two bolts until they release the motor bracket.

(2) Rotate the boom until it reaches the parked position.

(3) Retain the boom there with the captive pin.

Even in such a simple task, there is ample opportunity for things to go wrong:

- The satellite may not be stationary. It would be easier to work on a stationary satellite because then there will be no centrifugal forces that would tend to move the boom. In the plan for EVA

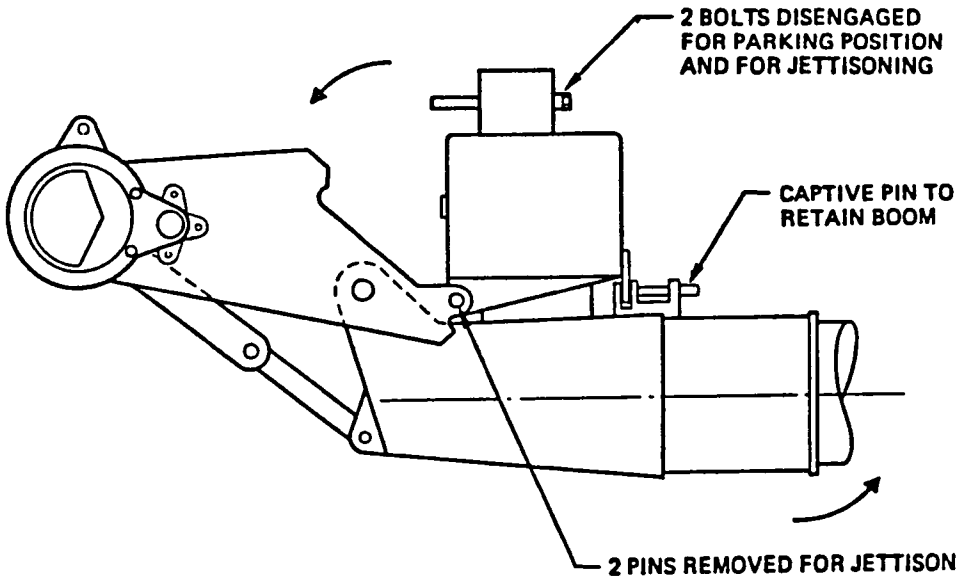


Figure 3: High Gain Antenna Boom (Parking Position for Servicing)

refueling, the RMS holds the satellite steady for the astronaut. In a plan for automated refueling, however, the telepresence system might be sent out to park the boom on the free-flying satellite. If it is completely out of fuel, its attitude control system will not be working properly, so there could be some residual spin The automation system would probably have to stabilize the GRO before it could stow the boom safely. It might measure the satellite's angular velocity vector, position itself nearby on the spin axis, match rotation rates, dock rapidly, and then use its own attitude stabilization system to de-spin the satellite. It may have to adapt to the new location of the center of gravity of the two linked machines to de-spin in the most fuel-efficient way. However, if the GRO has a standard grapple on its stable spin axis, the automation system may be able to dock with it without first matching spins.

- Some of the parts that the telepresence system must work on may be out of reach. If so, the automation system may have to move around the satellite as it works. In the boom-parking task, all the important parts—bolts, captive pin, boom, brackets, etc.—should be found within a working volume less than 1 meter across, and

they appear to be readily accessible.

- Moving some parts may take more force than the automation system can exert. This may be because something is stuck, or it may be because the automation system cannot get the needed grip. It is best if the automation system can clamp itself rigidly and strongly to the structure being worked on. Then, it can exert the full strength of its manipulators. If clamping is infeasible, then it may have to hold one part of the structure with, say, its left "hand" while it pushes or twists on another part with the right. Depending on the grip, however, this could place excessive strain on a joint in the left arm, and it may slip or even be damaged. The wrist joints are particularly susceptible, since a force applied at a large distance from the gripper on that wrist has a large lever arm.

- Objects may not be where they are supposed to be, or their precise position may not be known *a priori*. For example, the captive pin can probably move around rather freely when not screwed into the bracket. When redeploying the boom, the automation system may have to find the pin visually before inserting it through the clevis.

- It may not be possible to predict how often to repeat or continue an action. For example, suppose one of the bolts holding the bracket is difficult to turn for some reason—due to cold welding in the vacuum of space, perhaps. Then after running the screwdriver for the usual time interval, the screw may still be in the bracket. There are many ways to verify that a screwing or unscrewing operation has succeeded, of course—counting turns, monitoring torque, rotation rate, or travel, pulling on the screw afterward, visual inspection of head position, etc.

- An object may break when an attempt is made to move it. For example, one of the bolts may fracture (perhaps due to brittleness in the cold of space, or to sunlight-induced thermal strains). Depending upon when in the task this happens, and which screw it is, this may require an abort, a different method of releasing or securing the boom in position, or a minor change to the usual procedure.

- An object may move unexpectedly. For example, a thruster might fire accidentally on the GRO while the boom is being

placed back in its deployed position. The resulting inertial forces could cause the boom to swing around. If the motion goes undetected by the automation system, a collision could occur or a critical sensor could be occluded.

- Not all objects that the automation system has to handle will be floating freely. Some will be connected to other objects through mechanical linkages, so that they can only move along certain paths. The boom is a simple example—it can only move in a circle around its hinges. Trying to turn it with a very strong manipulator that has stiff position-control servos could break it off if the gripper does not move precisely along the correct circular path. In general, manipulating objects whose motions are partially constrained like this will require some "give" or compliance in the arm or the gripper. However, it should only comply in certain directions, and those directions may change as the gripper moves.

3. Use of a Telepresence System

In this section, we describe the use of a telepresence system to park the HGA boom on the GRO satellite. The physical capabilities of the telepresence system are described in Section B-3-a. In Section B-3-b, we explain how the operator would use these capabilities to park the boom. Finally, in section B-3-c, we present the sequence of actions that the operator would make the telepresence system perform during the boom-parking task.

A remotely controlled manipulator could be used with a mockup of the satellite to determine how difficult the boom-parking task would be, what kind of end-effectors would be most useful, how the time it takes to perform the task varies with servo stiffness, etc. Since the boom mechanism is essentially planar, it could be mounted with the axis of the boom hinge vertical to eliminate gravitationally induced torques and simulate a weightless environment.

a. *Built-In Capabilities*

The telepresence system would have the following built-in capabilities:

- Visual sensing—Remote viewing of the work area through television cameras.

- Tactile sensing—Includes the force-reflection capability as well as proximity sensing and sensing of pressure distributions over the surface of a gripper with a tactile array. The latter is an area of research that has not received much attention, due to scarcity of

high-resolution tactile sensing arrays. Now that these are becoming commercial products, the ability to feel the shape of an object in contact with the gripper or fingers of the slave may be a very valuable capability. Some experiments have been performed in presenting this sort of information tactilely by means of an array of small tactile stimulators (e.g., air jets [1]) held against the operator's fingertip.

- Operation of the sensors—Primarily, this means operating the cameras. It includes positioning the cameras, focusing them, adjusting their apertures, selecting lenses and filters, and assigning camera images to display screens. The other sensors in the system will have comparatively fewer controls, if any.

- Motion of the OMV free-flyer—This should include at least the ability for the operator to fire the thrusters remotely, using the television images from the ROSS for guidance However, an inertial navigation system on board the OMV would make possible automatic piloting and station-keeping capabilities as well. This would reduce the piloting skill requirements for an operator.

- Motion of the ROSS' arms—This is accomplished by means of the force-reflection system, which makes the slave arms follow the motions of the master control arms precisely, as long as they encounter no external resistance. The same force-reflection system also provides a capability for controlling the force and torque that the slave arms exert.

- Operation of the grippers on the ROSS' arms—This should at least include the ability to open and close a parallel-jaw gripper. More complex end-effectors could also be provided, with, e.g., force reflection, multiple articulated fingers, tool turrets, etc.

- Operation of the docking/rigidizing equipment—The simplest example would be operation of a standard docking grapple like the one on the Shuttle RMS. This only involves two types of action: open/close the grapple wires and rigidize/unrigidize. More general docking/rigidizing equipment is desirable so that the telepresence system can work anywhere on the space station, without requiring the presence of a docking probe there. It may not be necessary to introduce additional controls for such equipment. The slave arms could be used to attach it to

structures, etc.

b. *Task-Specific Control Inputs*

Sticking friction in the slave arm mechanism and, to a lesser extent, in the master controls, sets a lower limit on the forces that the operator can feel at the worksite through the force-reflection channel. Operators would probably develop strategies to overcome such limitations—for example, making small exploratory movements in different directions with the master controls in order to better feel where an object is.

Parking and redeploying the boom during refueling of the GRO satellite would probably not require a very dexterous gripper. A parallel-jaw clamp might be quite adequate, especially if it can hold a power screwdriver with which to loosen and tighten the bolts. Tactile feedback from the gripper itself may even be unnecessary, although visual feedback would then be more important so that the operator could see what the telepresence system was doing.

c. *Scenario for Boom Parking*

Docking/Undocking

Docking requires navigation to an accuracy of a few inches. RMS operators routinely "pilot" the end effector with this accuracy when grappling a payload. The RMS operator can judge the direction and amount of any misalignment by the appearance of a special visual target around the grappling hook as viewed by a camera mounted just beside the end-effector.

Similar techniques should be adequate for guiding a free-flying ROSS to docking with a satellite to be serviced. If the satellite does not have a docking grapple and target, the ROSS operator will have to judge the distance from the television image, laser rangefinder readings, or other means. Light-stripe illumination in conjunction with a transparent overlay on the tv monitor would be a simple and inexpensive method for measuring approach distance without using a computer.

Rigidizing/Unrigidizing

The purpose of rigidization is to attach the automation system firmly to the equipment it is working on so that it can exert forces without pushing itself away. It may be possible to carry out some low-force procedures with both the telepresence system and satellite flying free. NASA has designed various kinds of reactionless tools for astronauts that the ROSS could use as well. The ROSS should probably attach itself to the strong berthing adaptor framework that NASA plans to add to the GRO satellite to absorb umbilical mating forces. Some possible rigidization mechanisms include the following:

- A special bracket that the ROSS can attach to the adaptor, to which it can then attach itself.

- Special flexible articulated structures on the ROSS that can be attached to three or more points on the GRO and then made rigid. Several commercial devices could be adapted for this purpose.

- Tension cables that can be attached to six or more widely separated points on the GRO. By pushing itself away from the satellite, perhaps with a specialized actuator, the ROSS would tension the cables, and they would then hold it rigidly in position with respect to the satellite.

The rigidization equipment should probably be unpowered, since the ROSS operator can attach it to the GRO with the slave arms.

Parking/Redeploying the Boom

Redeploying the boom would require more precise manipulation than parking, because it involves starting two bolts in threaded holes, while parking only requires insertion of a pin. Getting the screwdriver into the heads of the screws could be easy or difficult, too, depending upon the type of screw head used. Bondhus Balldriver screw heads may be a good choice, since they reputedly allow a larger angular misalignment between screwdriver and screw than slotted, hex, or Philips heads.

Positioning the boom should be relatively easy, requiring only simple arm motions and small forces, unless the hinge mechanism is jammed. The operator is more likely to damage the boom if he grasps it firmly with the gripper than if he just pushes the boom with it, for two reasons. First, the gripper could damage the boom by squeezing it too hard. Second, if the gripper does not have sufficient compliance in the right directions, and the arm is strong, it could bend the boom or tear it out of its hinges. High-quality force reflection would prevent this, of course, as would the programmable compliance capability that JPL is developing. Pushing the boom would be safer, because the boom is not squeezed by the gripper. Furthermore, if the gripper does not move along the precise circular path required, it will simply slide along the arm instead of exerting large forces.

4. Use of an Adaptive Robot

In this section, we describe the use of a sensor-equipped, computer-controlled adaptive robot to park the HGA boom of the GRO satellite. In Section B-4-a, we describe the physical characteristics of the adaptive robot. In Section B-4-b, we describe a set of built-in capabilities supported by the robot's control software. In Section B-4-c, we explain how the robot's programmer would use these capabilities. Finally, in Section B-4-d, we present the sequence of activities that the adaptive robot would carry out during the boom-parking task.

a. *System Description*

An adaptive robot differs from a telepresence system in that the human operator is replaced by a computer. The role of the human shifts from continuous "hands-on" control to (1) an initial programming effort, (2) deciding when the robot should operate, and possibly (3) supervising its progress.

A telepresence system, such as the ROSS, BAT, or TWS could form the basis for an adaptive robot. Using proven electromechanical equipment would reduce development risk, time, and cost. A remote control capability is desirable in any robotic system for maintenance and emergencies. Evolving a robot from a telepresence system would provide it "for free." Some minor equipment changes would probably have to be made, such as reducing slack or friction in the mechanical parts to improve manipulator performance, or replacing the television cameras with automation cameras that interface to a computer more easily. Unfortunately, the ROSS is still only a design concept, and the BAT and PFMA have not been able to take advantage of the most modern materials, motors, computers, programming techniques, or control theory to produce the highest-performance teleoperator for space [2, 3].

It will probably be quite practical to provide substantial local computational power—on the order of a VAX-780 or a cluster of MC68000's—aboard an adaptive robot, so that it can operate relatively autonomously of the space station computer system. Power, packaging, and mass storage requirements would seem to pose the main difficulties. The tradeoffs between centralized and distributed control should be looked at closely.

An intelligent control system for a robot manipulator must be able to plan ahead, know when it has to change the posture of the arm, and make sure that the change will not interfere with the task being performed. The control system would have to be able to decide what to do when a particular motion is impossible. In the case of a free flyer, for example, it might re-orient the flyer on which the arm is mounted.

b. *Built-in Capabilities*

Generally, an adaptive robot should be able to adapt to minor variations in the positions of objects in its environment. More specifically, it should be able to do the following:

- Determine its own position—e.g., by using navigation satellites, vision, inertial guidance, or other means.
- Locate a known object using vision, touch, or other senses.
- Adjust its actions to match the actual positions of the objects it works with.
- Sense the position and velocity of its own articulated structures such as arms, fingers, legs, feet, "necks," etc.
- Compute the actuator positions necessary to position or move its articulated structures in any specified way.
- Sense forces and torques exerted on its body and limbs, especially on its grippers or other end-effectors.
- Transform positions, velocities, forces, torques, and other spatial quantities from one reference frame to another fast enough to support real-time control of arms, legs, cameras, etc.
- Exert force and torque on an object in arbitrary directions with its limbs.
- Adjust the effective compliance of its limbs to suit specific task requirements.
- Sense how it is grasping an object and control its gripper to maintain a firm grasp. This may involve interpretation of high-resolution tactile array images or finger-joint torques.

The examples above represent four points along an evolutionary path in the direction of increasing automation of the robot-programming process. At each step, the computer takes over more of the responsibility for knowing how the automation equipment (the adaptive robot and its controlling computer) works. The result is the same in each of the first three stages—a program that makes the robot carry out a specific space station procedure. Such a program can be reused many times, and copies can be made to operate multiple robots simultaneously or

at different places on the space station. In the fourth stage of direct control by an AI program, a complete, "stand-alone" robot control program of this sort might never be created for a task, any more than people plan every action they will take during the day. However, if an intelligent robot should develop a widely useful procedure, it could easily broadcast a program-like description of it to other similar robots.

The first stage in the above list represents current practice in the robot industry. The second stage is the subject of much research at present, and isolated parts of the problem seem to be solved, or are close to solution. The third and fourth stages are long-term goals for artificial intelligence research. An important pacing technology for the latter two is geometric reasoning, based in part on CAD data about individual part shapes [4]. Another is automatic planning, based on knowledge of how the parts function in space station equipment.

Lately, pursuit of semiautomatic program generation is causing a trend away from *symbolic programming** and towards *procedural programming.*† In symbolic programming, the programmer *tells* the computer (in a formal programming language with a well-defined syntax and grammar) what the robots should do and how they should do it. In procedural programming, the programmer *demonstrates* to the computer what the robots should do. He or she may operate simulated robots on a graphic display or use the real ones. By means of menu picks or other simple interactions, the operator tells the computer the meaning of each action he makes the robots perform. For example, the purpose of a particular arm motion might be to point out a position in which to place a workpiece, to describe a tool trajectory, or to define a gripper orientation for grasping a particular object.

c. *Task-Specific Control Inputs*

NASA and its contractors typically plan activities to be carried out in space in great detail, and they have developed formal methods for describing them (e.g., mission plans). This sort of information provides an excellent starting point for generating programs to control adaptive robots for the same activities. Some of the kinds of information these plans contain that would translate easily into elements of a robot program are the following:

---

*Not to be confused with *symbolic processors*, computers with a special architecture for artificial intelligence

†The VAL™ system that controls Adept and Westinghouse robots, and PLACE™ by McDonnel-Douglas are good examples of commercial procedural programming systems

- Step-by-step sequences of activities to perform.
- Tests to make.
- Explicit criteria for judging the test results and deciding what to do next.
- Fix-up procedures to perform when a test indicates a problem.
- Abort sequences to perform when an activity must be abandoned.
- Final functional tests to verify that the purpose of the activity has been achieved successfully.

To turn such activity descriptions into a robot program, one must ask how a robot can use its mechanical sensors and effectors to mimic what a person would do with his organic ones. Prior experience with teleoperators in space will provide much of the knowledge we will need to program robots effectively.

Many robotic systems already outperform people in strength, endurance, and the ability to carry out complex or repetitive procedures without error. They can remember precisely and indefinitely the location of any object they see, so they may only need to look at a scene once to carry out a task. A modern robot can tell rather precisely where its hand is, and where an object in its field of view is. So, in some tasks in which a person would have to spend a long time making careful measurements with special instruments, a robot might be able to make the same measurements much more quickly and with no extra equipment. Future robots systems will surpass people in additional ways—they may have ten arms and eyes in their fingertips, for example.

In the preceding section, we mentioned three milestones in the automation of programming adaptive robots. We will treat the second of the three here, in which the programming system takes the responsibility for routine coding in the robot programming language, and the person is merely responsible for specifying the individual arm actions, sensing activities, and decisions to be made.

The person requires two main skills in this situation: familiarity with the capabilities and limitations of the available robots and an understanding of how the boom mechanism works and what can go wrong with it. He might decide to describe first the parking and redeployment procedures under the assumption that no problems take place. Then he could go back over his instructions and expand the program by inserting verification tests for foreseeable problems and an appropriate fix-up or abort procedure for each.

In describing how to release the bracket by loosening the bolts, for example, the robot programmer might know from the standard mission plan or from physical intuition that the two bolts must be loosened first. Then he would use his knowledge of the robotic equipment, physical intuition, and perhaps a few computer-generated graphical simulations to decide whether the particular robot could loosen the bolts with its gripper alone, or whether it would need to use a tool, and if so, which tool.

Let us say he decides that the robot ought to use an electric screwdriver. Next he might realize that the available screwdriver takes several different types and sizes of bits. The bit would have to match the heads of the bolts, and he might ask the computer to ask the space station design data base what kind of heads the bolts have. The robot might have a tool carrier, so he would then select a place in the carrier for the screwdriver. Next he might program the unscrewing procedure by interacting with a graphic simulation of the task. He might use joysticks, keyboards, voice input, or other modern input-output equipment to describe how the arms should move to obtain the screwdriver from the tool rack, attach it to one of the arms, and position the tip of screwdriver near the bolt.

At this point, it would probably be necessary to describe a visual inspection procedure to locate the head of the bolt. Again graphic simulation would be useful in displaying what the robot's camera might see from different viewpoints with different types of imaging (e.g., binary, gray scale, color, or three-dimensional imagery). The programmer might select three-dimensional imagery and then specify a good camera position from which to look at the bolts. At this point, he might decide which of several cameras in the system to use. If he decides to use a camera on the end of one of the arms, he would then describe an approach path to avoid the other arm and the various parts of the GRO satellite. Finally, he might tell the robot to try to match the geometric model of the bolt head to the image, and that it should expect to find exactly two of them within a few millimeters of where the satellite's CAD data base says they should be.

Returning to the main sequence of the program, he would then specify an approach position for the screwdriver tip relative to the observed position of the bolt head (which the real robot would fill in later after it takes a picture of the real bolt heads). Then he would specify a straight-line move to mate the screwdriver tip with the head. He might decide to use remote-center compliance at the tool tip to prevent jamming or wedging during the mating operation. If so, he would have to specify the compliance parameters such as spring constants and the location of the center of compliance relative to the screwdriver tip—e.g., by simply pointing at that location in the image of the screwdriver on the graphical display.

Next, he would have to decide how to make sure the bolt is completely loosened. He might decide to simply run the screwdriver for a fixed length of time, say 10 seconds. He would then tell the programming system to insert a 10-second delay in the program between starting and stopping the screwdriver.

It may seem that the programmer in this scenario is having to attend to an unreasonable amount of detail for such a simple activity as unscrewing a bolt. In fact, the method of working described here would be a considerable *improvement* in terms of programming convenience over current robot programming practice! In this scenario, the programming environment is relieving the programmer of even more tedious efforts that now must be carried out by hand, such as the following:

- Constructing a syntactically-correct program several hundred lines long (for a task of this complexity).
- Naming about two dozen coordinate reference frames and remembering how they are defined in terms of one another.
- Measuring distances and angles on the satellite in order to establish nominal values for those frames in terms of Cartesian coordinates and Euler angles.
- Coding data-base access requests to the space station data system for the necessary CAD data.

d. *Scenario for an Adaptive Robot*

The ROSS, operated as an adaptive robot, would fly on the OMV from the space station to the satellite and back. It would use on-board navigation equipment until it was in the vicinity of the work area. Then it would use visual navigation for terminal guidance, matching a CAD model of the work area to what its television cameras could see to determine its position. A three-dimensional imaging system would be particularly useful for terminal guidance to avoid obstacles that are not found in the model.

Next it would locate attachment points visually. The simplest case would be to locate a docking probe which is in a known location on the satellite. More generally, it might look for places on the satellite to which it could attach its rigidization equipment. An intermediate approach would be to provide inexpensive "hard points" on the satellite. These could consist of a simple post or hole, or a simple mechanical coupling that mates with corresponding couplings on the robot's rigidization equipment.

Next, the robot would locate the various components of the boom joint mechanism. The adaptive robot would make heavy use of its visual and tactile sensors to accommodate small deviations in the positions of the objects it works on. The motions with which it is preprogrammed would be expressed in terms of positions and orientations relative to the actual observed locations of these objects. Motion commands expressed in terms of absolute positions are inappropriate because of inevitable deviations of the position of the workpieces and the robot itself from their nominal positions as they would be given in a CAD data base. If the CAD data base can also provide information about the limits of such variability, it could be used to verify that objects in the view of the cameras have been correctly identified. An object that seems to be very far from its expected position may have been misidentified. If not, something else may be wrong, such as structural damage to the equipment. An adaptive robot would be able to recognize that such a situation exists. However, it would not usually be able to reprogram itself sufficiently to adapt to a problem such as gross structural damage, and would probably have to abort its mission.

A two-armed adaptive robot would park and redeploy the boom in much the same way as a person would, using its arms to manipulate a screwdriver to loosen and tighten the screws, to move the boom from one position to another, and to insert and remove the locking pin. It would be much easier for the robot to use a power screwdriver than a "manual" one, though this is not out of the question. A robot can have more arms than a person, however. A two-armed robot is probably the most convenient for the boom-parking task (one arm to deal with fasteners, the other to handle the boom), but multiple arms should prove very useful in more complicated tasks.

5. Use of an Intelligent Robot

In this section, we describe the use of an "intelligent" robot to park the HGA boom of the GRO satellite. The robot's physical characteristics and a set of built-in capabilities supported by the robot's control software are described. We then explain how a crewperson would use these capabilities, and indicate the sequence of activities that the "intelligent" robot would carry out during the boom-parking.

a. *Description*

An artificially intelligent robot might look very similar to the adaptive robot or telepresence system and indeed might be developed from them. The main physical difference would probably be a much more powerful on-board computer system (assuming it is not remotely controlled by the central space station computers). Functionally, it would be able to program itself to a much greater degree than the adaptive robot programming system in the preceding example. It

would also require considerably more sophisticated built-in capabilities that, in fact, far exceeed the present state of the art.

b. *Built-in Capabilities*

An artificially intelligent robot would be capable of the following:

- Situation Assessment—The ability to deduce from sensory observations and previous knowledge the important facts about its surroundings. It would have to be able to deal with incomplete or even contradictory information.

- Automatic Planning—The ability to devise a complex schedule of activities in order to accomplish a particular mission. This would involve reasoning about resources, conflicts, and plan reliability. An advanced topic is how to insert sensor actions partway through a plan to obtain the information needed to complete it.

- Plan Execution and Monitoring—Comparing the current situation to the situation anticipated in the plan, noting any problems, and utilizing any unplanned-for advantages that occur. It also involves deciding whether a plan has gone so far wrong that it is necessary to revise the plan.

- Automatic Replanning—Generating a new plan to suit the present circumstances. This is different from the original planning problem, because (1) it may have to be done rapidly, and (2) much useful information will have been generated during the original planning process that may be of use.

An important issue in all of the above is the computer representation of various kinds of knowledge. The representation must be concise so that much information can be made available to a program, yet it must also allow any information likely to be needed to be computed or deduced easily. Some particularly important kinds of knowledge for space station applications are part shapes, degrees of freedom of motion of parts in mechanism, tool capabilities, customary procedures and functions, common-sense knowledge about matter, energy, time, and causality, and the method of operation of the space station equipment itself.

c. *Task-Specific Control Inputs*

Compared to the previous example, the information that the person would have to supply to an "intelligent" system is negligible. One might only have to tell it in English that the boom must be placed in the parked or deployed position. Alternatively, one might graphically depict the desired position of the boom via a

simulation on a display screen. The robot's AI software would then figure out how to accomplish the task.

d. *Scenario for the Intelligent Robot*

In parking the boom, the "intelligent" robot would first collect what knowledge it has or can obtain from the space station data bank relating to the GRO satellite, the task, the current situation on the space station, available resources, and so on. Then, like a person, it would probably run geometric simulations ("visualize") how the task might be performed, note the equipment it would need, and make up a tentative schedule. It would then obtain any tools and supplies it decided it needed and set out for the satellite at the right time.

On arrival, it would inspect the GRO to verify that its actual condition matched its own assumptions in all important respects for the plan it had made. For example, it might find that the two retaining bolts have already worked themselves loose. In that case, it might decide to amend the plan by omitting the bolt-loosening activities.

It would be a more serious problem if the bolts are jammed. The robot would then have to devise an alternative method of parking the boom. It might use its understanding of mechanisms and its ability to reason about geometrical relationships to deduce that removing one of the pins in the toggle linkage and the boom hinge pins would allow the boom to be moved to the parked position and secured.

The above operations imply that the robot is able to sense and interpret the world, reason about geometric and temporal relationships and constraints, and reformulate is plans when the world turns out not to be as it had expected. Much research and development effort is required to attain these robotic abilities, but the advantages of such a system over adaptive robotics and telepresence are obvious. It would be much more reliable than an adaptive robot because it would be much more resourceful. It would be much more productive than telepresence, because so little human time would be spent in performing a given space station task

REFERENCES

1. J.W. Hill and J.C. Bliss, "Tactile Perception Studies Related to Teleoperator Systems," (Final Report 2), Menlo Park, California, SRI International, June 1971. Prepared for NASA Ames Research Center, Moffet Field, California, under NASA Contract NAS2-5409, SRI Project No. 7948.

2. H. Asada and H. Yamamoto, "Torque Feedback Control of MIT Direct-Drive Robot," *Proceedings of the 14 th International Symposium on Industrial Robots*, pp. 663-670, Göthenburg, Sweden (2-4 October 1984).

3. R.S. Wallace, "Robotics and the Space Station," *Computers in Engineering 1984, Advanced Automation: 1984 and Beyond, Volume One*, pp. 369-373, Las Vegas, Nevada (12-15 August 1984).

4. "Representation and Processing of Spatial Knowledge," (Computer Science Technical Report TR-1275), College Park, Maryland, Computer Science Department, University of Maryland, May 1983. Sponsored by NASA Grant NAG5-290

Appendix C
ADVANCED MEMORY TECHNOLOGIES

C. ADVANCED MEMORY TECHNOLOGIES

Appendix C

ADVANCED MEMORY TECHNOLOGIES

We consider briefly the state of the intensive research efforts in the U.S. and Japan on advanced optical, magnetic, and semiconductor memory techniques and their impact on memory-system architecture for the Space Station Infomation System.

1. Memory Technology

a. *Optical Memory*

Very high capacity read-only optical memories for both video and digital information are available commercially. Some laboratory versions of write-once optical memories have been demonstrated, but the major target of research in optical memories has been the read-erase-write store that can be cycled without limit. The desired functionality has been demonstrated successfully for several years, but with unacceptable limitations of various kinds.

Recent reports indicate that the goal of a practical system is in sight*. Strong commercial incentives should yield working systems within three years. Accurate capacity estimates are not available, but, as a point of reference, fixed video disks contain about 1.5×10^{10} individual spots (capable of storing about 20,000 letter-size page images). Assuming 1:1 correspondence of spots to bits, this implies a capacity of about 2,000 Mbytes (current high-performance winchester magnetic memories have about 200 Mbytes capacity). Data rates of 50 Mbytes/sec are quoted for some optical disks.

b. *Magnetic Recording*

A major development in magnetic recording is the use of perpendicular recording (magnetization normal to the recording surface rather than longitudinal, as in current practice), which is predicted to result in a ten-fold increase in linear density. This is accompanied by significant advances in the recording medium and in recording-head design (e.g., three-fold density increase for each) [3]. Motivating forces for these developments include the normal competition among producers of

---

* [1] describes work by 3M Co , Sony, Canon, IBM and Fujitsu, reported at the January meeting of the Intl Society of Optical Engineering; [2] reports on claims by Hitachi Research Laboratories for a non-destructive, high S/N technique.

office-type memories (floppy disks and winchester drives), but there is also an awareness of the impending competition from reversible optical memories.

c. *Drive Problems*

Almost all optical and magnetic-surface memories employ rotating media. Rotating magnetic memories use the fluid properties of air to achieve close control of head spacing. It will be necessary to compensate for the inertial effects of the rotating mass of the recording medium on space-station orientation and to provide for a safe supply of air to assure safe spacing of the recording head under loss of ambient air. These appear to be solvable problems for the space environment, but we are not aware of the current state of space-qualification for rotating magnetic memory systems. Head-space maintenance for optical recorders would not seem to be a significant problem. Some experimental developments exist in which a read-only optical recording medium is stationary, while the light beam is scanned by a rotating mirror. This would be advantageous for the space environment, but it is not clear that manufacturers will follow this path.

d. *Bubble Memory*

Magnetic bubble memories have had a difficult history of commercialization. Of the seven major manufacturers who invested in the technology, four remain active, and new products are being announced currently (e.g., a 4-Mbit chip by Intel appears to be forthcoming). These may provide a useful combination of low cost, reliability, and access speed in Space Station computing (at one time, NASA considered bubble memories as potential replacement for tape recorders, which have had a traditionally poor record of reliability in the space environment).

e. *Semiconductor Memory*

Semiconductor memories continue their well advertised advances in density. One-megabit chips will be available in quantity within three years, and four-megabit chips will follow within another three years. The sensitivity of memories with very small bit-cell areas to the level of radiation in the space environment deserves careful attention. Transient disturbances can be dealt with by error-correcting coding and the use of nonvolatile storage for backup (with appropriate architectural mechanisms), but the possibility of permanent damage may be a serious problem. These concerns may delay the introduction of the very highest density chips in space applications.

2. Architectural Implications

These technical developments suggest that there will be very good capacities for mass storage of data in the form of high-density optical and magnetic memories, and, furthermore, that the protocols for use of mass memories will be more-or-less conventional, e.g., data will be erasable and recordable in blocks and will have access latencies in the low tens of milliseconds.

Memory hierarchies will be needed to fill the range of computing requirements for access speed and capacity, but no major disparities in capacities or speeds appear likely, given the continuing advances in all forms of memory technology. Careful architectural planning (size balancing, channel bandwidth, etc.) will be needed to assure a smooth flow of data between levels of the hierarchy, together with adequate levels of protection for data against environmental and internal sources of error and loss of data.

REFERENCES

1. "Optical Recording Looks to New Frontier," *Electronics Week*, February 4 1985, pp. 24.

2. "Color-Memory Alloys Promise Cheap Erasable Laser Discs," *High Technology*, March 1985, 5(3), pp. 7.

3. "Magnetic Mass Storage Densities Rise," *Electronics Week*, October 29 1984.

Appendix D

COLLINEAR HIERARCHICAL DECOMPOSITIONS

D. COLLINEAR HIERARCHICAL DECOMPOSITIONS

Appendix D

COLLINEAR HIERARCHICAL DECOMPOSITIONS

1. Introduction

The basic problem addressed here is how the various requirements can be fulfilled, while at the same time developing a system concept that respects the notion of a hierarchy of successive criticality, suitably defined. Thus, we seek the creation of a single hierarchy that encompasses all of the different notions of criticality in order to provide an ordering that permits cleaner design, less complex implementation, easier maintenance, and less likelihood of deleterious changes compromising the criticality requirements. This approach can be looked at as an attempt to map all of the criticality requirements onto a single conceptual architectural structure, which perhaps (but not necessarily) is a hierarchy.*

2. A Hierarchical Approach to Criticality

A fundamental problem in computer systems used to control critical environments (such as the Space Station) is that they must satisfy a variety of critical requirements simultaneously. Above all, such systems must provide adequate support for any life-critical operations. The desired system safety in turn requires a wide range of characteristics, typically including (among others): sufficiently reliable hardware; fault tolerance (presumably in hardware and in software) to mask hardware faults; reliable (although not necessarily totally correct) software; security (e.g., user privacy, system integrity, and data integrity). There are also other attributes on which critical behavior may depend, such as rapid maintainability of the system in response to unforeseen emergencies. Furthermore, even a sound design can be seriously compromised by a slightly defective implementation. For example, timing dependencies may become critical, particularly in distributed systems. A common problem is the occurrence of totally unanticipated system deadlocks that are extremely difficult to diagnose, and from which real-time recovery may be very difficult or impossible. Two particularly valuable illustrations of unanticipated disasters are provided by the synchronization problem before the first Space Shuttle [1] and the complete collapse of the ARPANET [2].

---

*For the hierarchicalists among the readers, it need not be a totally ordered hierarchy, but could be a partially ordered domain structure

Not only is the running system of concern, but also its use, administration, and maintenance. Then, the entire environment in which the computer systems operate may also be critical; the interactions between the environment and the computer systems are often a source of serious problems, as are the people involved. Furthermore, each change made to a system may potentially alter its behavior adversely. Security, privacy and integrity issues are closely related to reliability and fault tolerance. In addition, although issues of human safety overlap somewhat with the notions of security-critical and fault-tolerance-critical hardware and software, they require more than just the combination of those requirements. A system that is unreliable can not be considered secure when operating in an unreliable mode, and most likely should not be considered safe; a system that is unsecure can not guarantee service—particularly if it may be easily crashed—and therefore also may not be safe. Safety requirements thus seem to result in a much wider set of weakest links than would be found in a system designed (just) to be secure or reliable.

Recent work has been directed at hierarchical system design decompositions based on *layers of abstraction* that represent *layers of downward-only functional dependence.* Each successive layer generally provides functionality that is more readily useful to the higher layers—and eventually by the ultimate user.*

In multilevel secure systems, the lowest layer may consist of a security kernel (implemented in hardware *and* software) that provides some basic notion of security, upon which are implemented trusted processes, then nonsecurity-critical operating system functions, then programming languages, then system-provided application subsystems, then customer-provided environments, and only then possibly user software. Each layer refines the facilities of the lower layers, and may provide fine-grained protection mechanisms more closely suited to the specific applications. As another example, some systems are structured according to a linear hierarchy of layers of protection enforced by the hardware, e.g., Multics, SCOMP, iAPX 286 and 386, and the Data General 10000 series, all of which use a ring-structured architecture.† In fault-tolerant systems, similar layerings of functionality are found. Basic hardware may be configured into a fault-tolerant system kernel upon which successive layers of system and

---

*See Reference [3] by David Parnas, who has considered various relations with which it is meaningful to talk about hierarchies, e g , module A uses module B, or module A depends for its correctness on module B, or module A calls module B. The point here is that two different notions of hierarchical ordering must be effectively combined into a single notion, respecting both design integrity and criticality.

†It should be noted that the ring structure is used not only to provide isolation of users from the system, but also to protect the system from itself—increasing its fault tolerance

application code are built—relying on the fault tolerance of the kernel. Each successive layer again refines the facilities of the lower layers, and provides fault tolerance more closely suited to each applications. However, both in security hierarchies and fault-tolerance hierarchies, violations of the hierarchical ordering can occur as a result of unforeseen events. Perfect designs and implementations and sufficiently malfunction-free hardware are generally assumed. Any deviation from those assumptions can generally undermine the entire system.

Somewhat different conceptually is the notion of *degrees of criticality*—although in some cases the degrees of criticality can be related to the layers of abstraction and layers of protection. For example, Multics has a most critical innermost layer whose malfunction can cripple the entire system, then layers whose malfunction can cripple only the executing user process, then layers whose malfunction merely results in the user process returning to command level. (In Multics, these layers are in fact supported by the concentric rings of protection in the hardware, so that there is some—but only superficial—relationship between security and reliability.) However, in most conventional systems, little attention seems to have been paid either to criticality or to abstraction and the consequences can be frequent system malfunctions, fault recovery that is very costly (in personnel, time, and/or unrecoverable data), security vulnerabilities, and general doubts as to system performance in an unanticipated emergency.

3. Criticality

We next consider various hierarchies illustrating differing degrees of criticality, as well as some design decompositions of systems and networks, illustrating different layers of design. We then examine whether life-critical, fault-tolerance, and security requirements can be applied within a common design hierarchy. (By then, the reader will already have noted an intended relationship between the degrees of criticality and the layers of abstraction.)

- Table D-1 is derived from Asimov's Three Laws of Robotics [4]. These laws have been useful in writing science fiction, but are also of interest as an illustrative (albeit superficial) model of a human-safety requirement, stated as follows:*

  (1) A robot may not injure a human being, or through inaction, allow a human being to come to harm.

---

*These three laws do not cover conflicting orders from competing humans, but might be extended to do so An analogy with the mandatory security policy implied by multilevel security is evident, where discretionary access is not covered by the simple model

(2) A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

(3) A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Four degrees of criticality are indicated in the table. The most critical (from a human point of view, at least) is the violation of the First Law, the next most critical being the violation of the Second or Third Law. Note that behavior satisfying all three laws is not necessarily correct, as illustrated by the distinction between degrees 2 and 3 in the table.*

- Table D-2 gives five illustrative hierarchical degrees of survivability, 0 to 4, with successively less serious effects resulting from hardware or software problems (hardware faults, design errors, etc.) associated with the execution of code in the particular layer of the system. For example, the Multics, SCOMP, iAPX 286/386, and Data General 10000 ring-structured hardware directly supports this kind of a decomposition based on survivability. Note that designs of most conventional systems are not generally sufficiently decoupled to guarantee that most faults can be sufficiently isolated.

- Table D-3 shows a similar illustrative decomposition into four degrees of network criticality. One might suspect that in a well-designed network, no malfunction of a node or of the network could be able to render the entire network useless. However, such a case actually occurred in the ARPANET collapse of 27 October 1980, in which two spurious status messages propagated and caused the entire network to become flooded by highest-priority status messages that rapidly dominated all network traffic [2]. This event effectively ground the entire net to a halt, as well as rendering it unmaintainable through the normal maintenance interface (from BBN in Cambridge using the net itself). The challenge in network design is to minimize or possibly eliminate the chance of such a disaster.

---

*For more technically motivated recent work in modelling safety properties, see References [5] and [6]

- Table D-4 shows six illustrative degrees of criticality with respect to the impenetrability (and penetrability) of a centralized system.

- Table D-5 shows four degrees of generic criticality in an expert system. This table is highly suggestive of the extremely difficult problem of designing and implementing an expert system so that its behavior remains controllable despite a variety of potential problems, such as (a) subversion by exploitation of underlying operating system problems to change the expert system itself, (b) incompleteness of the rule base, (c) subversion by adding a contradictory rule, and (d) subversion by adding a rule of higher priority. An extremely interesting research question is whether layered structure in the design and perhaps design verification (e.g., of sufficient completeness) can contribute to intrinsically safe expert systems.

4. Design Hierarchies

Before attempting to consider whether these degrees of criticality might be respected by a common design that accommodates all of the requirements at once, consider first several examples of design hierarchies. (Not surprisingly, security is often a concern of these systems; the most extensive use of hierarchical decompositions to date has come from the security community—but then only from the viewpoint of security. However, that work has some significant impact on the other criticality requirements considered here.)

- Table D-6 shows a typical architectural decomposition of a multilevel secure (MLS) system, based on a security kernel and some trusted subsystems that together are responsible for the maintenance of multilevel security separation (i.e., no adverse flow of information, downward to a lower level or laterally to another compartment at the same level). Typical *security levels* are Unclassified, Confidential, Secret, and Top-Secret, and should not be confused with the *layers* discussed here.* Penetration or malfunction associated with each layer has a corresponding effect.

---

*Although the levels of multilevel security are presumably not needed in the unclassified SSIS, there are several reasons for mentioning MLS here. One is that it illustrates the importance of having a mandatory policy for required behavior that must not be compromised—whether it is a notion of security or fault tolerance or timely, life-critical performance A second is that the kinds of compartmentation available in MLS may indeed be of great value in the SSIS, although all of the categories could be at an unclassified level. Thirdly, the notion of multilevel integrity (noted below) is of significant relevance in its own right, and can be better understood in the light of multilevel security.

However, note that a user or a malfunction could compromise the entire system from an outer layer, if the design and implementation are not sufficiently correct (e.g., including security and fault tolerance of both software and hardware). In other words, a single weak link—a flaw in the design, or a hardware malfunction—could invalidate the validity of the isolation of critical functionality, irrespective of the layer in which the running program was executing. Thus, the strict isolation of the hierarchical design is a hope, but by no means a certainty—even with extensive analysis, formal verification, or whatever. SCOMP [7] and KSOS [8] are two examples of systems supporting multilevel security.

- Table D-7 shows a similar decomposition, based on a kernel that enforces multilevel integrity (MLI). Integrity is generally thought of as a formal dual of security—although the names of the *integrity levels* are different from the names of the security levels. Integrity levels might be called Low-Risk, Medium-Risk, and High-Risk, implying a measure of trust associated with (say) a program or piece of data. Implementation of integrity-level separation can be used to prevent tampering with the system by less trusted individuals—and in combination with multilevel security can ensure the absence of Trojan horses, viruses, etc., that violate the MLS/MLI requirements.* Considering the criticality of the applications, some form of system and data integrity is recommended for the SSIS.

- Table D-8 shows the hierarchy of the UNIX United system—including its Newcastle Connection [9]—a collection of UNIX systems are organized into a network in which all communications among the different UNIX systems are provided by the Newcastle connection. The appearance of a single file system is provided, although it is implemented across all of the constituent UNIX file systems. Although the individual UNIX systems are not able to enforce rigid security partitions, the Newcastle Connection is.

- Table D-9 shows the abstract-machine hierarchy of the SRI design

*Note that MLS by itself does not prevent Trojan horses that cause unauthorized altering of critical information, and that MLI by itself does not prevent unauthorized reading of information in violation of MLS

for a Provably Secure System (PSOS) [10] that is based on a hierarchy of type managers. Each type manager is responsible for the management of its particular class of objects or resources (including whatever protection, fault tolerance, and distributed implementations are required). Protection is based on the use of nonforgeable objects called *capabilities* that are used to control access to all objects (not just memory). In this system, a different policy can enforced by each type manager, relevant to the objects that it manages. (For example, the user object level must guarantee that no objects will exist that can no longer be accessed [i.e., lost objects]. The extended type layer must guarantee type safety of its types.)

- Table D-10 gives an illustrative hierarchy in which each layer may have different fault-tolerance measures applied, and in which different functionality may be distributed.

- Table D-11 shows the same illustrative hierarchical decomposition, with different security techniques applicable at each layer. Again distribution is possible at various levels.

- Table D-12 shows the ISO Open Systems Interconnection (OSI) network hierarchy. This network hierarchy was established largely oblivious to the security problems; there are indeed serious problems with respect to security. We suggest at each layer [called a level in OSI parlance] some of the potential security problems that need to be addressed.

These hierarchical decompositions are presented here primarily for illustrative purposes. The main point is that a system design must consider *all* of the relevant critical requirements simultaneously, including its application requirements. Furthermore, the design and implementation for any critical requirement (e.g., fault tolerance and security) can generally be distributed among different layers in the design and among different components in its (distributed or centralized) implementation, according to where the various mechanisms are most needed and can be most effective.

What we are seeking is a single consolidated hierarchy that encompasses all of the requirements. For simplicity, we refer to this as a *collinear* hierarchy, in that it would be a projection of a system and its applications onto a single design decomposition. We must address whether or not such a hierarchical design concept is realistic. However, we note that even the single-purposed hierarchies can be compromised by certain events not covered by the assumptions under

which they are designed.

To amplify the notion of interactions among the requirements, consider the following.

- A system that is not secure is not likely to be reliable, for it may be easy for an intruder to crash it, or for the system itself to violate its own self-protection.

- A system that is not reliable is not likely to be secure, for its behavior under fault modes is unknown—and potentially nonsecure.

- A system that is either nonreliable or nonsecure is not likely to be able to fulfill its life-critical requirements.

- A system that is not safe to use (in the life-critical sense) could still appear to be secure and fault tolerant, although those attributes might then be irrelevant.

Thus, these requirements are closely interrelated.* The resulting conclusion with respect to a hierarchy of criticality *and* overall functionality is suggested by Tables D-13 and D-14. Table D-13 represents a conventionally designed system. Here there is no sharply delimited criticality kernel or explicit criticality hierarchy that is reflected in the system design. In this case, any hardware "burp" or program flaw—even in supposedly system-independent application code could cause a disaster such as loss of life, crash of the system, total deadlock among parts of the system, or collapse of the system security.†

Table D-14 represents a generic system in which the criticality hierarchies and the design hierarchies (e.g., for fault-tolerance, security, life-criticality, and evolution) are collinear. In this case, only those problems that directly affect the kernel should be able to cause disaster—and those problems should have been covered defensively or minimized extensively by good design (including extensively fault-tolerant hardware and software), careful implementation, and thorough

*For example, see [11] for a discussion of some of the interactions between security and reliability.

†Note that building a supposedly secure and reliable application subsystem on an underlying operating system [hardware and software] that is neither secure nor reliable is very dangerous. As an example, consider MVS, which is extremely weak with respect to security. For this reason, SKK's ACF-2 and IBM's RACF were developed However, although IBM systems running either ACF-2 or RACF on top of MVS appear superficially to be secure against penetration, they in fact do not stand up against even casual attack. For example, see [12].

verification.

In order to minimize the effect of uncovered hardware malfunctions, and to provide some of the isolation of critical functionality from less critical functions, a distributed architecture seems highly advantageous, organized according to the criticality hierarchy. In this case, it might appear that the collinearity could be relaxed among different components, e.g., if one component is security critical and another is life-critical. But this ignores the practical problem that the different requirements may interact with one another, as noted above.

It is important to recall that excessive modularity can be harmful. A common first reaction to the notion of a hierarchical design is that it *obviously cannot be implemented efficiently.* This is indeed true if implemented blindly, with many nested layers of interpretation. However, with appropriate design and suitable hardware, efficiency need not be an obstacle. (In the PSOS design of Table D-9, for example, a single hardware instruction can be used to implement a procedure call at layer 12, even though procedures at lower layers may be implemented in software!)

It is also important to note that without suitable hardware support, a hierarchical approach can be very costly. However, in order to satisfy the critical requirements in the future, more sophisticated hardware and software will be needed.

5. Model Hierarchies

Desired design properties may be associated with each layer in a hierarchical design Although each of the above designs could be so considered, the reader is by now already exhausted. Thus, just three model hierarchies are illustrated here to illustrate the nature of relevant properties.

- Table D-15 gives examples of properties appropriate at each layer in the combined design hierarchies of Tables D-6 and D-7. Note that the SCOMP system has undergone formal proofs that its kernel (layer 0) satisfies a formal MLS model (except for the trusted processes and some storage channels) and is undergoing further proofs that its trusted processes (layer 1) satisfy the same model.

- Table D-16 gives examples of properties appropriate at various layers of the design.

- Table D-17 gives examples of various layers of proof properties associated with the design of a Software-Implemented Fault-

Tolerant system (SIFT) [13]. Some proofs have been carried out for a paper version of the SIFT system that is running at NASA Langley [13].

Various safety properties can be considered in a similar vein. Neither Asimov's nor Leveson's (References [5] and [6]) were conceived within a hierarchical framework, rather being properties of the user interface. As such, they are useful starting places for the hierarchicalization of requirements for human safety.

Although the presentation here is extremely sketchy, significant background is needed to make these examples meaningful to the uninitiated reader. However, the main point to be made is that layers of design hierarchy can be related to degrees of criticality with the help of models of explicit properties that need to be satisfied at each layer.

6. Conclusions

This discussion represents a small step toward the design, development, and modeling of particularly critical systems that recognize the importance of a variety of critical requirements.

Satisfying a set of criticality requirements is a holistic system problem, and requires an overall design approach that respects all of the requirements. The approach of a collinear hierarchy that respects all of the requirements appears to be useful. Otherwise, problems left unsolved may ultimately have catastrophic effects, e.g., hardware malfunctions and software bugs, with unsafe behavior, security flaws, and system collapses as possible consequences. Unless significant structure is present in the design, such problems may propagate wildly. However, even in a highly structured system or a system whose distributed implementation provides strong isolation to reduce propagation, unforeseen failures can cause disaster (as in the case of the ARPANET collapse).

Thus, this is only a first step. More work is needed to make these concepts precise, to address the many lurking problems, to model the desired safety requirements, and to examine whether the concepts are realistic.

Table 1: Criticality: Human and Robot Survival

Effects of Robot Faults (after Asimov)
3. Completely safe *and* correct behavior
2. Satisfaction of all three laws, providing safe but not necessarily correct operation
1. Injury or death of robot (violates Law 2 or 3)
0. Injury or death of human(violates Law 1)

Table 2: Criticality: Survival of Computing Service

Effects of HW/SW Faults
4. No adverse effect
3. Can abort command, leave process intact
2. Can crash one user process
1. Can crash one user process group
0. Can crash entire system

Table 3: Criticality: Survival of Network Service

Effects of HW/SW Problem in Distributed Net
3. Loss of one packet; if retry successful, no visible effect
2. Loss of one link between two nodes, reroute
1. Loss of one node and comm to attached hosts; reroute for hosts having multiple nodes
0. Disruption of entire network (e.g., ARPANET collapse)

Table 4: Criticality: Immunity to Penetration

Effects of HW/SW Vulnerability
5. No adverse effect (perfect system!!)
4. Penetration of one resource (file, server)
3 Penetration of one user login only; one user compromised, others may be also, due to OS flaws and sharable resources
2. Penetration of an entire user group (RWE)
1. Penetration of every user's login/files (RWE) (e.g., by accessing unencrypted password file)
0. Penetration of system databases (RW); forced crashes (E)

Table 5: Criticality: Effectiveness of Expert Systems for Critical Functions

Effects of HW/SW Faults in a Structured Expert System
3. Extended rules cannot compromise critical behavioral requirements, but can be changed for experimental purposes.
2. Primitive rules sound, sufficiently complete for critical use.
1 Expert system kernel itself is sound, will satisfy critical behavior requirements, including its nonsubvertibility, irrespective of expert knowledge given to the system.
0 Underlying operating system is unsound, and can be used to compromise the expert system.

Table 6: Criticality: Multilevel-Secure (MLS) Computings

Hierarchy of MLS Service Functions and Effects of Compromise
4. End-user interface (restricted set of user functions); breaking restricted shell should not compromise MLS!
3. Exported vendor interface (user profiles, limits); vendor-imposed discretionary security subvertible
2. Exported application interface (subsystems); application subvertible, but not MLS compromise.
1. Restricted trusted subsystem interface (privileges masked); trusted subsystem MLS subvertible (e.g., login, printer spooler), possibly whole system.
0. Privileged MLS kernel interface; total MLS compromise possible.

Table 7: Criticality: Multilevel Integrity (Trust)

Hierarchy of Multilevel Integrity and Effects of Compromise
4. End-user interface; compromise of vendor integrity
3. Exported vendor interface; compromise of application
2. Exported application interface; application integrity violated, but not MLI compromise
1. Restricted trusted subsystem interface; Trojan horses installable that violate MLS and MLI.
0. Privileged integrity kernel interface; total MLS/MLI compromise.

Table 8: Process Abstraction in the UNIX United System

3. Users and user programs
2. UNIX operating system, with a distributed MLS file store
1. Newcastle Connection: providing trusted communications among different UNIX systems
0. UNIX kernel for each UNIX system

Table 9: Function Abstractions in PSOS (Provably Secure Operating System)

Layer	PSOS Abstraction or Function
16.	user request interpreter *
15.	user environments and name spaces *
14.	user input-output *
13.	procedure records *
12.	user processes*, visible input-output*
11.	creation and deletion of user objects*
10.	directories (*)[c11]
9.	extended types (*)[c11]
8.	segmentation and windows (*)[c11]
7.	paging [8]
6.	system processes and input-output [12]
5.	primitive input/output [6]
4.	arithmetic, other basic operations *
3.	clocks [6]
2.	interrupts [6]
1.	registers (*), addressable memory [7]
0.	capabilities *

* = functions visible at user interface.
(*) = partially visible at user interface.
[i] = module hidden by layer i.
[c11] = creation/deletion hidden by layer 11

Table 10: Virtual-System Hierarchy for Fault-Tolerance

Visible functions	Hidden functions	**Protocols.** *Distribution* and fault-tolerance mechanisms
Application	Other applications	**Application distribution protocols.** *Data distribution* (redundant or not), application-particular rollback.
Virtual network	Network ops, comm nodes	**Node-to-node protocols.** *Redundant network distribution*, alternate routing.
Virtual system	Other subsystems	**Subsystem-to-subsystem protocols.** *Cooperating subsystems*, isolation, data security, system integrity.
Virtual data bases	File formats, remote access	**Data query and retrieval protocols.** *Distributed data bases*, safe update strategies.
Virtual process	Process scheduling	**High-level interprocess protocols.** *Replicated processes*; *independent alternative processes*; *hidden process directories*; automatic rollback.
Virtual input-output	Buffering, Asynchrony	**I/O device protocols.** *Local net remote I/O.* Reliable controllers; safe asynchrony, handshaking;
Virtual file system	Protected directories, backup dumps	**File transfer protocols.** *Replication of critical data*, e.g., on different media; file archiving, rollback, and retrieval; safe updates.
Virtual memory	Storage addresses, retrieval	**Interdevice protocols.** *Shared disks*, *local-network remote access*; reliable address calculation.
Virtual uniprocessing	Multiprogramming	**Low-level interprocess protocols.** Process isolation (e.g., domain switching); safe interrupt mechanisms.
Multiprocessing	Processor coordination	**Interprocessor protocols.** Redundant interprocessor communication.

Table 11: Virtual-System Hierarchy for Security

Visible functions	Hidden functions	*Distribution* and security measures
Applicat-tions	Other applications	*Program, data distribution. ISO 7.* Trojan horse detection/elimination.
Virtual network	Network ops, comm nodes	*Hidden remote operations, data. ISO 6?* Encryption.
Virtual system	Other subsystems	*Cooperating subsystems. ISO 5?.* Subsystem isolation, mutual suspicion, confinement.
Virtual data bases	File formats, remote access	*Data queries and retrievals. Distributed databases.* Encryption, protected views.
Virtual process	Process scheduling	*High-level processes.* Hidden process directories. Process isolation.
Virtual input-output	Buffering, Asynchrony	*I/O devices.* Masking of real input-output, channel addresses.
Virtual file system	Protected directories, backup dumps	*Multifile systems.* File protection. Newcastle Connection Distributed Multitree.
Virtual memory	Storage addresses, retrieval	*Distributed memories.* Total encapsulation of real memory, memory protection.
Virtual uniprocessing	Multiprogramming	*Low-level interprocess protocols.* Processor isolation and controlled communication.
Multiprocessing	Processor coordination	*Hardware protection mechanisms,* Internal isolation, e.g. rings.

Table 12: ISO OSI Reference Monitor Hierarchy for Network Data Transfer
Open Systems Interconnection Architecture (ISO OSI RM)
[See ISO/TC 97/SC 16 N1643 October 1983 for Newer Proposal]

Level	Typical protocols	Some security problems
7. Application	IPC, EFTS	User behavior, unencrypted headers
6. Presentation	FTP, Telnet, Mail Virtual terminal, Job Transfer	Operating system flaws Remote maintenance spoofs Field encryption here?
5. Session	Ports, sockets, SNA, Courier (Xerox)	Input-output flaws
4. Transport	TCP, NBS/ECMA, ISO transport	End-to-end protocol bugs IEEE, transport encryption
3.5.Internet	IP	Weak gateways and nets
3. Network	X75/X.25(13)	Unprotected comm links
2. Link	HDLC(ISO), Ethernet, CMSA/CD contention, X.75/X.25(L2), Sytek token passing, Cambridge ring, BISYNC & SDLC (IBM), ADCCP (ANSI)	Contention, saturation, ring breaks, denial of service
1. Physical	ISO 2593, X.21, RS-232C & 449 (EIA)	Piggybacking, radiation

Table 13: Criticalities: Conventionally Designed System

Effects of HW/SW faults on criticality
2. Noncritical functions; total disaster possible
1. Less critical functions; total disaster possible
0. Most critical functions; total disaster possible

Table 14: Criticalities: Hierarchically Designed System

Effects of faults on criticality *and* functioanlity
2. Noncritical functions; disaster extremely unlikely, assuming reasonable hardware fault tolerance
1. Somewhat critical functions; sharply limited disaster, assuming reasonable hardware fault tolerance
0. Most-critical functions; disaster always possible but highly unlikely—with kernel extensively verified, and hardware reasonably fault tolerant

Table 15: Model Hierarchy: MLS/MLS

4. End-user interface: discretionary separation of users, application environment properties
3. Exported vendor interface: vendor-specific properties
2. Exported application interface: application-specific properties
1. Restricted trusted subsystem interface: e.g., no adverse flow at interface
0. Privileged Kernel interface: MLS/MLI with privilege mechanisms

Table 16: Model Hierarchy: PSOS

Layer	PSOS Abstraction and Model Property
n.	user-created type manager: soundness
15.	user spaces: search-path flaw avoidance
12.	user processes isolated, I-O sound
11.	user objects: no lost objects
10.	directories: type safety, no wrong dirs
9.	extended types: generic type safety
8.	segmentation: virtual memory gives proper segment, no residues after deletion, paging (layer 7) hidden
6.	system processes: interrupts, I-O masked
4.	basic operations: correctly implemented
0.	capabilities nonforgeable, nonbypassable, nonalterable

NOTE: Honeywell's Secure Ada Target embeds the MLS property into the capability mechanism.

Table 17: Model Hierarchy: SIFT

```
                SIFT Model Hierarchy
-----------------------------------------------------
                                                -10
Markov Model    Failure Probability 10    /hr
  /             using HW error-rate analysis
 /
( I/O Model     System SAFE =>
 \  |           ''all tasks correct''
  \ |
Replication     Task replicated;
  Model         Values voted upon on
    |           task completion
    |
 Activity       Task actitivies: startup,
  Model         broadcast of values, vote
    |           execute, synchronization
    |
 Operating      SPECIAL specs for OS:
  System        scheduler, voter, dispatcher
    |           buffer manager, etc.
    |
  Pascal        Pascal code for each routine
 Programs
    |
BDX-930 Code
```

REFERENCES

1. J. Garman, "The Bug Heard 'Round the World," *ACM SIGSOFT Software Engineering Notes*, October 1981, *6*(5), pp. 3-10.

2. E. Rosen, "Vulnerabilities of Network Control Protocols," *ACM SIGSOFT Software Engineering Notes*, January 1981, *6*(1), pp. 6-8.

3. D. L. Parnas, "On a 'Buzzword': Hierarchical Structure," *Information Processing 74 (Proc. IFIP Congress 1974)*, pp. Software: 336-339 (1974).

4. I. Asimov, "Runaround," *Astounding Science Fiction*, April 1941. Anthologized in "I, Robot" and "The Complete Robot".

5. N.G. Leveson, "Software Safety in Computer Controlled Systems," *IEEE Computer*, February 1984, *17*(2), pp. 48-55.

6. N.G. Leveson et al., "Design for Safe Software," *Proc. AIAA 21st Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Reno NV. (January 1983).

7. L.J. Fraim, "SCOMP: A Solution to the Multilevel Security Problem," *IEEE Computer*, July 1983, *16*(7), pp. 26-34.

8. E. J. McCauley and P. Drongowski, "KSOS: Design of a Secure Operating System," *Proceedings of the NCC*, NCC '79 (June 1979). Reprinted in Rein Turn (ed.), Advances in Computer Security, Artech House, 1981.

9. J.M. Rushby and B. Randell, "A Distributed Secure System," *IEEE Computer*, July 1983, *16*(7), pp. 55-67.

10. P.G. Neumann, R.S. Boyer, R.J. Feiertag, K.N. Levitt, and L. Robinson, "A Provably Secure Operating System: The System, Its Applications, and Proofs," (Tech. Rep.), SRI International, May 1980. Second Edition.

11. B.W. Lampson, "Redundancy and Robustness in Memory Protection," *Information Processing 74 (Proc. IFIP Congress 1974)*, pp. Hardware II: 128-132 (1974).

12. Ronald Paans, Guus Bonnes, "Surreptitious Security Violation in the MVS Operating System," (Tech. Rep.), Delft University of Technology, Department of Electrical Engineering, POBox 5031, 2600 GA Delft, Netherlands, 1984.

13. P. M. Melliar-Smith and R. L. Schwartz, "Formal Specification and Mechanical Verification of SIFT: a Fault-Tolerant Flight Control System," *IEEE Transactions on Computers*, July 1982, pp. 616-630.

Appendix E

COMPUTER-COMMUNICATION SECURITY

E. COMPUTER-COMMUNICATION SECURITY

Appendix E

COMPUTER-COMMUNICATION SECURITY

Experience has demonstrated conclusively that security requirements for critical systems must be anticipated well in advance of the system development. This appendix is a preliminary analysis of the security and privacy requirements for the extensively automated space station. It contains a summary of generic security requirements, followed by an initial consideration of which of those generic requirements might be relevant to the space station. It also addresses system architectural issues arising from a preliminary interpretation of those requirements.

1. Generic Security Requirements

This section provides a summary of the generic requirements for security and data privacy potentially relevant to a particular system effort. It thus serves as a working checklist that should be applied to any particular system under study—whether for requirements definition or for prevention of penetration.

The summary is based on the results of the 1982 summer study of the National Academy of Science Sciences,.National Research Council, and Air Force Study Board, which focused on multilevel database management security. The final report of that summer study included a table of security issues that are of particular relevance to secure systems and secure databases. From that table we have derived Table E-1, to provide the basis for the present generic summary.

For the reader who is experienced in the security area, Table E-1 provides a relatively self-contained reminder of computer system security issues. For the reader who is not so experienced, the subsequent discussion provides an overview of the problems implicit in the table.

Authorization Policies

Security policies are of two basic types—those dealing with discretionary security and and those dealing with mandatory ("nondiscretionary") security. Examples of currently implemented discretionary security are access control lists (found in the Honeywell Multics and SCOMP systems, and in the Data General MV/10000 AOS/VS) and group-based controls (e.g., in UNIX and TOPS-20, with

AUTHORIZATION POLICIES—

Mandatory policies ("nondiscretionary"): *e.g.,* multilevel security with levels, compartments, and dissemination markings; access controls based on classification markings and user clearance.

Discretionary policies: access control lists, group-based access (*e.g.,* user/group/others), capability-based access, user profiles, authorization chains and multiple keys.

Disclosure policies: reading, exceptions, leakage, residues, sources; inference; classification by association, aggregation; sanitization: summarization, abridgment, misinformation; dissemination and distribution; manual and automatic downgrading.

Data integrity policies: alteration, deletion.

Changes in policy parameters and access rights, revocation of rights.

SYSTEM ISSUES—

Authentication: login, positive identification, initialization.

Auditing: detection of unusual events; individual accountability; noncircumventability and inviolability of audit trails.

Availability: denial of services (performance and other resources).

System integrity: controlling system changes; preventing spoofing from outside, implantation of Trojan Horses from within; viruses; trap doors; configuration management and maintenance.

Internal data migration: copying, archiving, dumps, recovery.

External data migration: sending data from one classification system or database to another; relabelling data with new classifications.

Mutual suspicion: sensitive programs operating on sensitive data.

OTHER ISSUES—

Communication security: encryption, key management.

Physical security: enclosures, guards, personnel procedures.

Operations security: reducing emanations, defusing traffic analysis.

Emergency actions: preparation for and response to forcible takeover and other critical events; recovery from intentional or accidental data destruction or system outage.

People: actions of system security officers, database administrators; models of human behavior integrated with policy models.

Derived from the Report of the National Academy of Sciences, National Research Council, Air Force Studies Board, 1982 Summer Study on Multilevel Database Management Security, Marvin Schaefer, Chairman), Chapter 3, D.E. Denning and P G Neumann eds.), 1983. Report FOR OFFICIAL USE ONLY

Table 1: Security Issues

protection according to user, group, and others). Less commonly found discretionary access policies involve capability-based access, access based on user-specific profiles, authorization chains involving a sequence of authorizations, and use of multiple keys to permit access.

The mandatory policy in common use involves the DoD classification levels [TopSecret, Secret, Confidential, Unclassified] and categories; the basic mandatory policy is that information shall not flow downward to a lower level or laterally to another category at the same level. (The situation is complicated somewhat in practice by the existence of dissemination markings (e.g., NOFORN or EYES-ONLY).)

Policies—whether mandatory or discretionary—deal with disclosure, alteration, and deletion of information, and with changes to the authorizations themselves. Disclosure policies must include not just the reading of information, but also exception conditions (from which much can often be gleaned), leakage through covert channels, residues left after incomplete deletion or deallocation, and protection of certain internal information that needs to be hidden totally from the ordinary user.

Such policies usually ignore—but in many cases should address—some second-order issues, e.g., the derivation of apparently hidden information by inferences. The inference problem is closely related to classification by association, in which the aggregation of data at a particular sensitivity level must be treated as more sensitive than any of the constituents. Disaggregation (the removal of a piece from an aggregate) can have the same effect. For example, the choice of a particular piece of an otherwise nonsensitive database provides an indication of interest in that piece—which itself may be a sensitive piece of information. (An example is given by a user's interest in a particular piece of unclassified weather data.) Other second-order issues include the downgrading of information, under a variety of circumstances, such as the sanitization of a sensitive document by summarization, abridgement, or production of misinformation. Note that the misinformation problem is compounded by the fact that information can be inferred by inconsistencies in the misinformation. (This is the problem of trying to lie consistently.)

One of the most difficult issues in maintaining security lies in controlling the changes in access rights, including the revocation of existing rights. In mandatory access controls, this is related to downgrading and upgrading. In discretionary access controls, revocation is particularly critical. (Note that read access cannot be revoked if past read access has been permitted, in the sense that a copy of everything could already have been made. However, revocation is meaningful with respect to blocking access to future changes.)

System Issues

The above issues deal largely with access rights (to data, procedures, resources, etc.). Also important are various system issues. A very significant problem is the authentication of users, i.e., ensuring that a user is really who he/she claims to be. Authentication of systems is also important in many applications, to prevent users being given the impression that a fraudulent system is actually the real thing (e.g., spoofing). Auditing of usage (especially detection and reporting of unusual events) also presents important security problems such as ensuring that the audit trails themselves cannot be compromised.

Reliability and high availability become security problems in the sense that a system needs to protect against unauthorized loss or degradation of resources. Unauthorized "denial of service" can become a serious security problem. Furthermore, an unreliable system may induce security flaws when it is in an unreliable mode of operation. Thus, there are critical interdependences between fault tolerance and security.

Other significant problems are: maintaining the integrity of the system as well as its data, preventing spoofing of the system by external users, and preventing the implantation of Trojan Horses by internal users. Thus configuration management and maintenance both impact the overall security. Detecting and controlling the spread of viruses is also a critical problem.

Other Issues

The above issues deal largely with the computer systems themselves. There are many other aspects of systems that influence security, including

- Communication security in protecting both global and local communications (e.g., through the use of encryption, protected wireways, and fiber optics), and associated problems such as key management.

- Operations security, in reducing the likelihood of electromagnetic emanations, external jamming, and deleterious traffic analysis.

- Physical security of the operating environment.

- Anticipation of responses to all sorts of emergencies.

Finally, improper behavior by the people who use and operate the system represent many different types of weak links; these include users, system security officers, database administrators, and maintenance and support personnel.

2. Security Requirements for the Automated Space Station

This section considers those generic security and privacy requirements that appear to be relevant to the automated space station. The present draft poses a number of questions that need to be answered through discussions involving SRI, NASA, and various contractors. Some can be answered by NASA, others will require some knowledge of the computer system design.

- General security questions: What security threats are considered to be realistic? Is everyone on-board equally trusted, or are some individuals more trusted than others? Will any national boundaries exist that will have to be reflected in the systems? Who will have access to the on-board computer systems? Will live reprogramming be prohibited, either locally (with expert programmers on board) or remotely? Is remote downloading (uploading?) of on-board programs intended? To what extent will direct control of the space station from the ground be permitted, e.g., in competition with local control?

- Mandatory policies: Will there be any needs for multilevel security, e.g., any need for classified data to coexist with unclassified data? It is important to anticipate such needs, rather than try to retrofit them at a later date. Such requirements impose very stringent constraints on the design and implementation of the computer systems and the communications. We assume that there will be *no* military multilevel security requirements within the computer systems, databases, and local networks within the space station computer complex, although we imagine (and assume in this discussion) that, on certain missions, there might indeed be classified computing performed through devices such as the PLIs on ARPANET/MILNET.

- Discretionary policies: What discretionary access controls might be needed? Are those that are generally available in today's commercial computer systems adequate? If not, what new features will be required? Are all users of the space station computers to be considered equally trustworthy? Are all data and programs considered to be equally sensitive, i.e., equally nonsensitive? We assume the coexisting presence of antagonistic forces, whether two competing on-board pharmaceutical manufacturers, or two rival nations.

- Disclosure policies: Will some outputs be more sensitive than others? Can any outputs from the computer systems be transmitted outside of the space station without human intervention? Are there any second-order security issues such as database inference, classification by association, or aggregation, as noted in Section E-1. If everyone on-board is equally trusted, there may be no such problems *internally*; however, these problems may still exist with respect to information leaving the space station. If everyone on-board is not equally trusted, it is very likely that such problems will exist in the on-board computers.

- Protection changes: Who will be authorized to make on-line alterations of database and system protection attributes? Can it be anyone? If not, then everyone on-board is not equally trusted. (Now reconsider your answers to the previous question.)

- Authentication: What kind of positive authentications should be required for individuals either on-board or accessing the computer systems remotely? (Spoofed communications from the ground could be a serious problem.) We assume that ground-based experimenters will want to interact on-line with their experiments, and suggest that nontrivial measures must be taken to prevent students and other random intruders from interfering therewith.

- Auditing: Are extensive audit trails to be maintained, similar to a flight recorder, but capable of recording all significant events? If, so, those audit trails must be noncircumventable and inviolable. Who might be permitted (read-only) access to the audit trails? Might anyone be authorized to *change* them? (Existence of such authorization would be very dangerous, and is strongly not recommended.)

- Availability: Is there any possibility that someone on-board could be a saboteur who might render certain services or certain data (e.g., controlling a particular experiment) unusable? (Commercial interests are particularly suspect here, although life-critical functions could conceivably be threatened for hostage or extortion purposes.)

- System integrity: Who should be authorized to make changes to the system or to the database structure? Is a threat perceived that the on-board computer systems might be spoofed externally

(from the ground, from foreign satellites, etc.)? What kind of manual configuration control will be necessary in times of system malfunction or partial outage?

- Internal data migration: Will on-line storage suffice, or will on-board off-line storage be required?

- External data migration: Will there be a need for different information sensitivities in data leaving the system? Will it be desired to protect return transmissions against hostile adversaries?

- Communication security: What threats must the data communications withstand, both internal to the space station and externally? What kinds of encryption gear will be permitted/required? Will keys need to be changed dynamically?

- Physical security: Is there any anticipation of physical intrusion, e.g., during a period when the space station is temporarily unmanned? Will there be physical compartments within the space station whose protection will reflect a need for different levels of physical security?

- Operations security: Is there any concern about enemy monitoring of internal computer and communication signals? Is there any sensitive information that can be derived from traffic analysis?

- Emergency actions: What emergency actions need to be anticipated?

- People: What different functions will space station occupants perform? What are the needs for differential treatment of these individuals?

End of Document