



1995 107772 G3/61 0027230

32704

SYMPOSIUM '92

OCTOBER 28-30, 1992
UNIVERSITY OF HOUSTON-CLEAR LAKE
BAYOU BUILDING



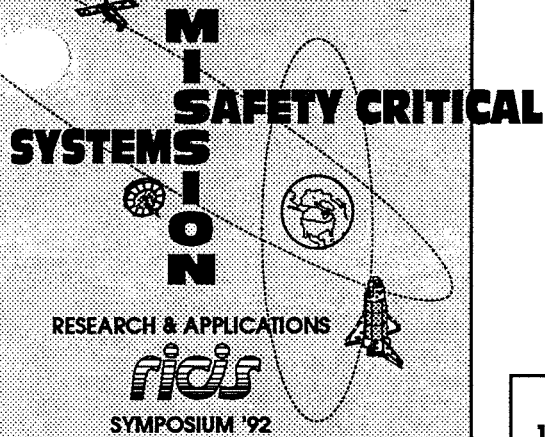
MISSION SAFETY CRITICAL SYSTEMS OPERATION



RESEARCH & APPLICATIONS



SPONSORED BY:
THE RESEARCH INSTITUTE FOR COMPUTING AND INFORMATION SYSTEMS
UNIVERSITY OF HOUSTON-CLEAR LAKE, BOX 444
2700 BAY AREA BLVD.
HOUSTON, TX 77058



RICIS SYMPOSIUM '92 SCHEDULE

(Gray pages for detailed descriptions)

**AUDITORIUM FOR ALL SESSIONS
WEDNESDAY, OCTOBER 28, 1992**

- | | |
|-------------|---|
| 1:00 - 1:15 | SYMPOSIUM WELCOME & INTRODUCTIONS |
| 1:15 - 2:00 | KEYNOTE ADDRESS |
| 2:15 - 3:30 | SESSION I: <i>SYSTEM SUPPORT FOR MASC APPLICATIONS</i> |
| 3:30 - 3:45 | BREAK |
| 3:45 - 5:00 | SESSION II: <i>GENERIC ARCHITECTURES FOR FUTURE FLIGHT SYSTEMS</i> |

THURSDAY, OCTOBER 29 1992

- | | |
|---------------|---|
| 9:00 - 10:30 | SESSION III: <i>REAL-TIME COMMUNICATIONS</i> |
| 10:30 - 10:45 | BREAK |
| 10:45 - 12:15 | SESSION IV: <i>SPACE STATION R&D AT RICIS - PANEL DISCUSSION</i> |
| 12:15 - 1:30 | LUNCH ATRIUM II |
| 1:30 - 3:00 | SESSION V: <i>LANGUAGE & SYSTEM STANDARDS</i> |
| 3:00 - 3:15 | BREAK |
| 3:15 - 4:45 | SESSION VI: <i>REAL-TIME ACTIVITIES - PETRO CHEMICAL INDUSTRIES</i> |
| 4:45 - 5:45 | WINE & CHEESE RECEPTION - ATRIUM I |
| 6:00 - 7:30 | BANQUET & SPEAKER - ATRIUM II |

FRIDAY, OCTOBER 30, 1992

- | | |
|--------------|---|
| 9:00 - 11:00 | SESSION VII: <i>FORMAL METHODS</i> |
|--------------|---|



**M
I
SAFETY CRITICAL
SYSTEMS
I
O
N**

RESEARCH & APPLICATIONS

ricis

SYMPOSIUM '92

1:00 - 1:15

SYMPOSIUM WELCOME & INTRODUCTIONS

*A. Glen Houston, Director, UHCL Research
Institute for Computing and Information Systems*

Paul J. Weitz, Deputy Director, NASA Johnson Space Center

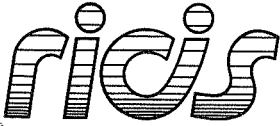
Glenn A. Goerke, President, Univ. of Houston-Clear Lake

1:15 - 2:00

KEYNOTE SPEAKER:

AARON COHEN

Acting Deputy Administrator, NASA Headquarters



Welcome to RICIS Symposium '92!

The University of Houston-Clear Lake's Research Institute for Computing and Information Systems is focusing its 1992 technical conference on Mission and Safety Critical Systems Research and Applications. This year's conference theme was selected on the basis of the ever growing importance of software for such systems throughout the world as nations move further into the automation age. Within the U.S., such systems are particularly important to NASA, the aerospace community, the petro-chemical and medical industries as well as to a myriad of other important applications.

In keeping with the mission of RICIS, to collaborate with industry, government and other academic institutions, the sessions and panel discussions have been organized to include distinguished professionals from these mutually supporting sectors. The sessions concentrate on supporting system research for: MASC Applications, Generic Architectures for Future Flight Systems, Real-time Communications, Space Station Research and Development at RICIS, Language and System Standards, Real-time Activities in the Petrochemical Industries and Formal Methods.

On behalf of our conference organizers, the University of Houston-Clear Lake, Texas A&M Computer Sciences Department, RICIS, all of our other partners and an outstanding group of presenters, we thank you for your interest in the research efforts and knowledge provided by this team, and for your participation in this symposium!

A handwritten signature in cursive script that reads "A. Glen Houston".

A. Glen Houston
Director, RICIS

A handwritten signature in cursive script that reads "R. B. MacDonald".

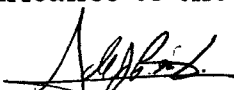
R. B. MacDonald
RICIS Technical Officer, NASA JSC

MISSION AND SAFETY CRITICAL SYSTEMS
Research and Applications
RICIS Symposium 1992

In the present "information age" that we have entered in the past few years, most every aspect of our lives is touched by a computing system in one way or another. Examples would include the appliances in our homes, the automobiles we drive, our schools and universities, hospitals, small businesses, the oil exploration fields and the military and space programs. These computer systems can be classified into two basic groups. One group of systems whose operation provide comfort and convenience and the other which is controlling systems whose failure to operate correctly could produce the loss of life and or property. The theme of this symposium centers on the second group, the Mission and Safety Critical Systems.

Some of the disciplines necessary for the realization of these systems are still evolving. That is why this symposium has a mix of formal presentations and panel discussions among established authors in the field. Our intention is to present a mix of results obtained from recent research and discussions of some of the open issues that must still be addressed.

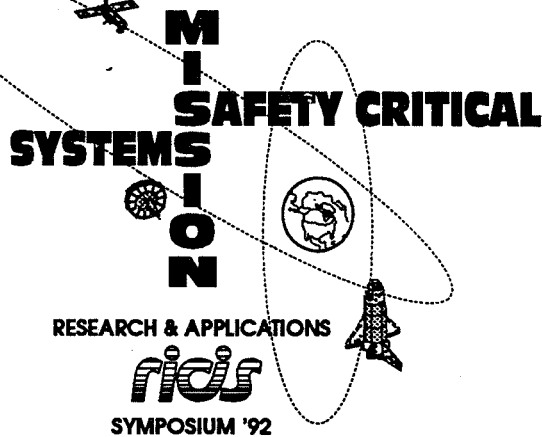
The symposium represents the dedication and hard work of many. We would like to thank all the people that in one way or another were involved with the organization of the symposium, for without their efforts this symposium would not have been possible. We thank the session leaders for their committment to arrange for the presenters and panelist's participation; Sandi Smith and Don Myers for the design of program announcements and day to day organization details; Robert MacDonald and Sadegh Davari who provided overall guidance and went out of their way when extra help was needed; and most importantly to the presenters and panel participants whose research activities and results constitute "the main event" of the symposium. Their recognized leadership and pioneering efforts provide valuable insights on the design of MASC systems. Their views on the strength and weaknesses of different approaches are of great significance to the scientific community.



Alfredo Pérez-Dávila, Symposium Co-Chair



Ted F. Leibfried, Symposium Co-Chair



Aaron Cohen
Acting Deputy Administrator, NASA Headquarters
Director, Lyndon B. Johnson Space Center

BIOGRAPHY

Aaron Cohen was appointed Director, Lyndon B. Johnson Space Center, Houston, Texas, on October 12, 1986. Mr. Cohen is responsible for directing a large and diverse work force of approximately 3,600 NASA employees and 14,000 support contractor personnel and managing an annual budget of approximately \$2.7 billion to accomplish the functions and programs of the Johnson Space Center. Mr. Cohen's primary management and technical responsibilities include: advancing the U.S. capabilities in space through the development and operation of the National Space Transportation System (Space Shuttle); managing major elements of the Space Station Program; developing and maintaining substantial capabilities in the fields of space systems engineering, mission operations, life sciences, research, lunar and planetary sciences, and Earth observations; maintaining a significant aircraft operations program; and selecting and training flight crews for NASA's manned space flight programs.

Cohen's career at the Johnson Space Center began in 1962 (at that time the Manned Spacecraft Center) in the Apollo Spacecraft Program Office. He managed the hardware and software designed to provide guidance, navigation, and control for both the Command and Service Module (CSM) and the Lunar Module. Cohen served as Chief, Systems Integration Branch in the Systems Engineering Division; Assistant Chief, Systems Engineering Division; and Chief, Command Service Module Project Engineering Division. From 1969 to 1972, Cohen was the Manager for the Command and Service Modules, Apollo Spacecraft Program.

As Director of Research and Engineering at the Johnson Space Center from 1983 to 1986, Cohen was responsible for the overall direction and management of all engineering and space and life science research and development in support of the major manned

space flight programs assigned to the Johnson Space Center. He was the center's Director of Engineering and Development from 1982 to 1983.

As Manager of the Space Shuttle Orbiter Project from 1972 to 1982, Cohen directed the design, development, production, and test flights (STS-1 through STS-4) of the Space Shuttle Orbiter to various line organizations within the Johnson Space Center, to other NASA centers, and to various contractors. As Manager and technical expert for the Orbiter, he maintained a continuous review of the technical aspects of the status of systems and related programs to ensure a balanced advance on engineering, scientific, and technical frontiers while managing the budget for the Orbiter Project.

Prior to these assignments, he was a microwave tube design engineer at RCA, where he developed patents on a microwave tube and a color TV tube, and a senior research engineer at General Dynamics Corporation.

Born on January 5, 1931, in Corsicana, Texas, Cohen received a Bachelor of Science in Mechanical Engineering from Texas A&M University in 1952 and a Master of Science in Applied Mathematics from Stevens Institute of Technology in 1958. He has taken advanced graduate study in mathematical physics at New York University and the University of California at Los Angeles. In 1982, he received an Honorary Doctor of Engineering from Stevens Institute of Technology and in 1989 an Honorary Doctor of Humane Letters from the University of Houston.

Cohen is a fellow of the American Astronautics Society and a fellow of the American Institute of Aeronautics and Astronautics. His NASA awards include two Exceptional Service Medals, two Outstanding Leadership Medals, three Distinguished Service Medals, and an Engineer of the Year Award in 1982. Other awards include Presidential Rank of Meritorious Executive for Senior Executive Service; Rank of Distinguished Executive for Senior Executive Service in 1982 and 1988; W. Randolph Lovelace II Award; President's Certificate of Recognition from the AAS; the AIAA Von Karman Lectureship in Astronautics; ASME Medal for 1984; Texas A&M College of Engineering Alumni Honor Award in 1987; UH-CL Distinguished Leadership Award in 1988; elected a member in the National Academy of Engineering in 1988; a joint recipient of the Goddard Memorial Trophy in 1989; the Distinguished Alumni Award from Texas A&M in 1989; the Gold Knight of Management Award, NMA, Texas Gulf Coast Council; 1990 Executive Excellence Award for Distinguished Executive Service, Senior Executives Association Professional Development League; and the 1990 National Space Trophy from the Rotary National Award for Space Achievement Foundation. Cohen has authored many articles for scientific and technical journals and publications and presented the Lawrence Hargrave Lecture at the International Aerospace Congress in 1991.

A veteran of the U.S. Army, Cohen and his wife, Ruth, are the parents of three children and reside in the Houston/Clear Lake area.

Remarks Prepared for Delivery:

Charting the Future

Challenges and Promise Ahead for Space Exploration

Aaron Cohen,
Acting Deputy Administrator
National Aeronautics and Space Administration

Research Institute for Computing and Information Systems
1992 Symposium
University of Houston-Clear Lake
October 28, 1992

Good afternoon, ladies and gentlemen.

I am very pleased to be with you here today for this symposium on mission safety critical systems. It is a topic that is near and dear to us at the Johnson Space Center, and I wish you well on navigating all the subjects on the agenda at this year's gathering.

I also am pleased to convey the Johnson Space Center's best wishes and a hearty well done to the symposium sponsor. The Research Institute for Computer and Information Systems has been a very valuable partner for JSC these last few years. The work that RICIS undertakes lies along one of the most dynamic frontiers in today's high technology world. There is virtually no element of our society, or our future, that is not touched in some way by the work done on the cutting edge of research in software and computers.

When RICIS was formed a few years ago, it helped fill a very important need for us. The Institute provided a gateway service, allowing us to interact more effectively and efficiently with the academic community. It also has proven to be a place where innovative work is done in that most innovative of disciplines: software production.

There are a number of ways in which this innovative approach has already paid off, and I would like to take a moment and cite but two examples.

A few years ago, a new computer language known as Ada came over the horizon, and the people at RICIS were instrumental in helping JSC usher in the new era Ada brought with it. Ada was more than just a computer language. It was a transformation. While the state of the art had moved forward in the computer field, the state of practice at JSC was lingering, and Ada helped us push it forward.

With Ada and the new thinking it fostered, we were able to leap into the latest generation of software engineering, and I would venture to say that we now rank with the world's leading producers of code. RICIS was there as we made that leap, and we are proud of our work together.

And the proof, really, is in the systems that we fly and train with. I would invite you to consider the software that drives the Shuttle's General Purpose Computers, or the Orbiter avionics system in general, or the trainers and operational data systems that we use here on the ground. Those are the kinds of systems you are here to discuss. They define systems that require software capable of operating in the most severe environment of all: an environment where people's lives are on the line, and where zero fault tolerance is simply the starting point for the kind of care we have to employ.

As we look into the future, similar care will be needed in designing software that can tend to Space Station *Freedom*, and perform the tasks necessary to keeping such a massive and complicated structure operating safely and effectively for 30 years in low Earth orbit. Here too, RICIS has played a role.

Through the development of Rate Monotonic Scheduling, we now have a system that will allow *Freedom's* computers to budget their time, to choose between a variety of tasks and decide not only which one to do first, but how much time to spend in the process.

So I am especially pleased that this year's symposium is focusing on the role of research and applications in mission safety critical systems. I can think of no application more important than the real-time systems we use in mission operations every day.

It is our bread and butter at JSC, and there will be even greater need--and greater challenges--for such systems in the future. After all, it is one thing to design software that can keep your people and equipment healthy in the relatively shallow seas of low Earth orbit...

But we intend to strike out for the Moon again, and after that, into the deeper waters of the Solar System, to Mars and beyond, and the work you are doing now is fundamental to achieving those broad goals. And I think that is where we must all begin to think soberly about both the promise and the challenge that lies ahead.

There is no question that the United States believes an important part of its future lies on the space frontier. We are a spacefaring nation, and we will continue to explore and utilize space. But there also is challenge ahead, and I think I can sum it up for you because it fits neatly on a single vu-graph.

The graph I have in mind shows three lines, and all three plot the progress of budget scenarios. The top line is steadily ascending off into the future, and it represents the cost of doing what we would like to do in the future of U.S. space exploration. The middle line also is rising, but much more modestly than the topmost plot, and it represents the budget runout on the programs we now have on the books at NASA.

The bottom line, however, represents a realistic and hard-nosed assessment of what we can expect in annual appropriations through the end of this decade. And the bottom line essentially runs parallel to NASA's current share of the Federal budget. The fact is, all of us are going to have to pay close attention to that bottom line in the years to come.

The end of the Cold War and the rising concerns over this nation's financial situation mean that all of us in the space business are going to have to develop leaner, tighter and more effective ways to do the work that we know is necessary.

The continuing national deficit means that dollars will be scarce in the future. International cooperation will become more of a hallmark of our business as budgets shrink and other nations continue to develop their capabilities for space exploration. At the same time, competition in the aerospace industry among many nations will tighten in the years ahead.

I tell you this because I know there is concern right now over budgets, and we--all of us--are going to have to face the near certainty of flat budgets at least for the next several years. Together, I think we can.

In the Shuttle program, for example, our priorities are evolving. Our overall intent is to fly at least 6 to 8 flights each year safely and successfully through at least the year 2005, and potentially for several years after that. Space Shuttle Orbiters could be flying well into the next century, and if you look at the service lifetime of such high performance machines as the SR-71, you see that there are precedents for that ambition.

So we see ourselves as being in this enterprise for the long haul, and although we have learned a great deal in 51 flights, there is much, much more that we can learn.

You can be a part of that by helping us to operate better and more efficiently, by helping us to design the continuing upgrades that will make the Shuttle a versatile and valuable resource for at least the next decade, and possibly for longer than that.

In the near term, we intend to drastically reduce Shuttle operations costs over the next several years, and we fully intend to get smarter and leaner and better. Better operating systems are one way to go about achieving those results.

There is another trend we've already touched on, and that is the internationalization of space flight. We are about to embark, for example, on an ambitious series of cooperative flights with the Russian Space Agency. Next year we will fly a cosmonaut on the Shuttle, and in the next two years or so we will dock a Space Shuttle with the Mir space station.

We are eager to begin this exciting new partnership. In a few weeks, two cosmonauts will arrive at JSC to begin training with the STS-60 crew. In the meantime, we have steadily strengthened the ties with the European Space Agency, with Japan and with Canada. We have flown crew members from each of those entities in just the last few months, carried several major payloads to space for European sponsors, and conducted a large number of experiments with the Japanese and the Canadians aboard the Shuttle. The third Canadian to fly aboard the Shuttle is in space even as we speak.

So while the United States alone will find it difficult to carry on our efforts at an increased level, I think you will see that more and more, the major spacefaring powers are going to be dealing with exactly the same set of conditions. Look for us to draw ever closer together in the future, to mount ambitious and historic missions together that any one of us would be unable to do alone.

As that is happening, don't forget the nature of the beacons that are drawing all of the space exploring nations out into the cosmos. The need to reap advances in communications and imaging and materials processing and medical technologies will be a very, very important part of what continues to drive us out into space. But there is more to be found on the frontiers of the future, and that is why I am confident that space flight will continue to challenge and inspire the best our nations have to offer.

Space offers limitless resources to a planet that in the next century is going to find an ever growing need for those resources. Future sources of energy, for example, will be harnessed from the Helium-3 that lies on the surface of the Moon, and from the bountiful output of the Sun. And ultra-vacuum and low gravity conditions will become ever more important factors in the development of new products and techniques. And understanding the mechanisms that drive our planet, and gaining a better insight into how we and our world evolved, will always drive us outward into space, because that is where we will find many of the answers we seek.

If you accept the notion that space flight is an inevitable part of our future, then you also must realize how important your work is to that future.

We should all be mindful of the extraordinary leverage that mission-critical software can have in the design and development of spacecraft, and in such factors as costs, schedules and the myriad other things that our program managers must keep track of.

Flight software was one of the driving factors in the development of the Space Shuttle, for example. You would be hard-pressed to find an area of hardware or flight technique development that was *not* influenced by the design of the Shuttle's data processing systems and the software that courses through it.

The same will be true for future spacecraft, whether they orbit the Earth or travel to the Moon or take colonists to Mars.

The future of our business holds incredible promise, but first we have to navigate some difficult times just ahead.

So my message to you today is to carry on, to innovate, to work even harder in your research and learning, and to hang on. We need you. We value you. We will try to help you as we move through the '90s. But it won't be easy, and the only way to get there is to go as a team.

Again, please allow me to express on behalf of all the people of JSC our best wishes for your pursuits, and our hopes for continued success in all your endeavors.

Thank you very much.

#

Session I

**SYSTEM SUPPORT
FOR
MASC APPLICATIONS**

2:15 - 3:30

Session Leader: Charles McKay
University of Houston-Clear Lake

Panelists:

**European Workshop Industrial Computer System's
Approach To Design For Safety**

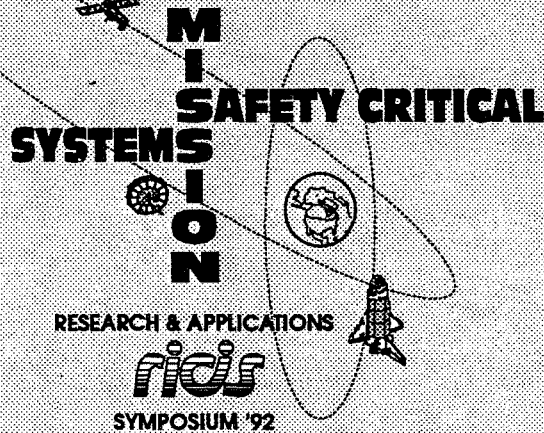
Janusz Zalewski, *Southwest Texas State University*

British Research

Alan Burns, *University of York, England, U.K.*

MISSION Research

Colin Atkinson, *University of Houston-Clear Lake*



SYSTEM SUPPORT FOR MASC APPLICATIONS

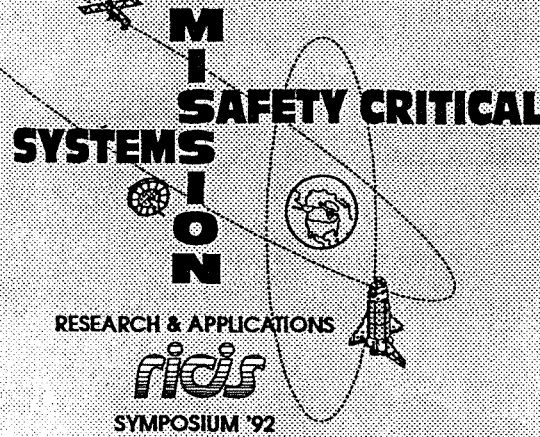
Charles McKay

Session Synopsis

This session is concerned with the problems encountered in the construction and maintenance of software systems whose failure could have catastrophic consequences. It will focus in particular on techniques, standards and guidelines which have been developed to improve the reliability of such "Mission and safety Critical" (MASC) systems, and to minimize the impact of those faults and failures which they are unable to tolerate. Included amongst the panelists are leading members of major projects from Europe and the US conducting research into MASC systems.

Biography

Dr. Charles McKay, founding director of the Software Engineering Research Center (SERC), is a professor of software engineering and computer engineering at UHCL. He is co-principal investigator, with Dr. Colin Atkinson, of the NASA-RICIS MISSION project concerning the development of systems software, and associated lifecycle techniques, to support the fault-tolerant execution of large, non-stop distributed systems. He also serves as chief scientist on the RICIS Repository Based Software Engineering (RBSE) Program and as chair of ARTEWG Interface Subgroup and Distributed Systems Task Force.

27-01
27231
p. 14
1995/07/7/3

387050

EUROPEAN WORKSHOP INDUSTRIAL COMPUTER SYSTEMS APPROACH TO DESIGN FOR SAFETY

Dr. Janusz Zalewski

ABSTRACT

This contribution presents a set of guidelines on designing systems for safety, developed by the Technical Committee 7 on Reliability and Safety of the European Workshop on Industrial Computer Systems (EWICS). Their focus is on complementing the traditional development process by adding the following four steps: (1) Overall Safety Analysis; (2) Analysis of the Functional Specification; (3) Designing for Safety; (4) Validation of Design; Quantitative assessment of safety is possible by means of a questionnaire composed of a number of modules covering various aspects of all major stages of system development.

BIOGRAPHY

Dr. Zalewski has been working for over 15 years in nuclear research institutes in Europe. As a member of the European Workshop on Industrial Computer Systems he participated in the development of EWICS guidelines for the construction of safety related systems. Most recently he has cooperated with the Data Acquisition Group of the Superconducting Super Collider Lab, in Dallas, working on the real-time kernels and real-time expert systems. Since 1989 he is on faculty at the Dept. of Computer Science, SW Texas State University, where he teaches Real-Time Systems and Software Engineering.

International Purdue Workshop

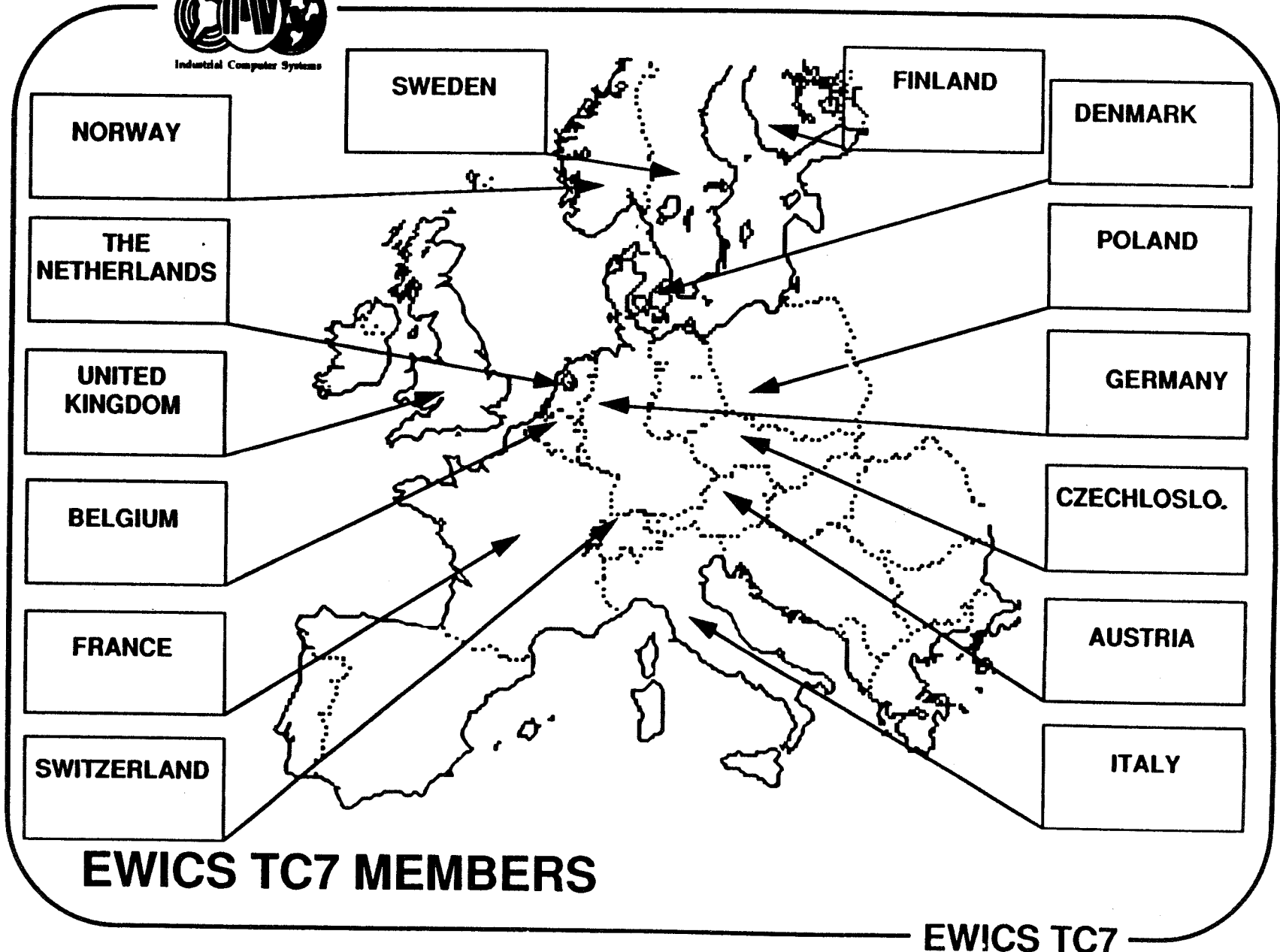


Industrial Computer Systems

EWICS APPROACH TO DESIGN FOR SAFETY

**Janusz Zalewski
Dept. of Computer Science
Southwest Texas State University
San Marcos, TX 78666-4616
JZ01@SWTEXAS.Bitnet**

EWICS TC7





MEMBERSHIP (Industries)

- **INFORMATION TECHNOLOGY**
- **NUCLEAR**
- **TRANSPORTATION**
- **ENERGY**
- **CHEMICAL**
- **TELECOMMUNICATION**

PRINCIPLES FOR SAFE DESIGN

- **RELATE TARGET SAFETY REQS TO PLANT SAFETY**
- **STRUCTURE ACCORDING TO CRITICALITY**
- **ULTRA-HIGH REL. OF SAFETY-CRITICAL MODULES**
- **DESIGN FOR SAFETY**
- **MONITOR SAFETY CONTINUOUSLY**
- **INDEPENDENCE OF SAFETY-RELATED ACTIVITIES**



FOUR SAFE DESIGN STEPS

- **OVERALL SAFETY ANALYSIS**
- **ANALYSIS OF THE FUNCTIONAL SPECIFICATION**
- **DESIGNING OF TARGET SYSTEM**
- **VALIDATION OF DESIGN**

1. OVERALL SAFETY ANALYSIS

- **INFORMATION OF THE ENVIRONMENT**
- **DESCRIPTION OF THE PLANT**
- **SAFETY CRITERIA**
- **REGULATIONS AND CONSTRAINTS**
- **AUXILIARY INFORMATION**
- **RISK ANALYSIS RESULTS**



2. ANALYSIS OF FUNCTIONAL SPEC

- **DECOMPOSITION OF THE SPECIFICATION**
- **CLASSIFICATION OF TARGET SYSTEM RESPONSES**
- **INVESTIGATION OF TARGET SYSTEMS INFLUENCES**
- **SAFETY ANALYSIS OF THE FUNCTIONAL SPEC**



3. DESIGNING OF TARGET SYSTEM

- **PRINCIPLES AND TECHNIQUES FOR SAFE DESIGN**
- **INTERFACES TO THE PLANT**
- **SPECIFIC DESIGN CONSTRAINTS**
- **SPECIFIC FEATURES TO ENHANCE SAFETY**
- **REVIEWED FUNCTIONAL SPECIFICATION**



4. VALIDATION OF DESIGN

- **CHECKING CRITERIA AND CONSTRAINTS**
- **FAILURE ANALYSIS**
- **FUNCTIONALITY CHECKS**
- **AVAILABILITY/MAINTAINABILITY CHECKS**
- **INTEGRITY/FAULT-TOLERANCE CHECKS**
- **EXTERNAL THREAT CHECKS**



THE QUESTIONNAIRE

- PROJECT PLANNING AND MANAGEMENT
- SYSTEM REQUIREMENTS SPECIFICATION
- DESIGN
- CODING AND CONSTRUCTION



THE QUESTIONNAIRE (Cont.)

- **INTEGRATION OF HARDWARE AND SOFTWARE**
- **VERIFICATION AND VALIDATION**
- **QUALIFICATION**
- **OPERATION AND MAINTENANCE**



SAFECOMP Workshops

- 1979 STUTTGART, GERMANY
- 1982 WEST LAFAYETTE, IN, USA
- 1983 CAMBRIDGE, UK
- 1985 COMO, ITALY
- 1986 SARLAT, FRANCE
- 1988 FULDA, GERMANY
- 1989 VIENNA, AUSTRIA
- 1991 TRONDHEIM, NORWAY
- 1992 ZURICH, SWITZERLAND

**F. Redmill (Ed.)
Dependability of Critical Computer Syst
Vol. 2, Elsevier, 1989**

**IEC STD 880, 1986
Software for Computers in the Safety
Systems of Nuclear Power Stations**



MISSION
SAFETY CRITICAL
SYSTEMS
ION

RESEARCH & APPLICATIONS

rics

SYMPOSIUM '92

British Research

Dr. Alan Burns

BIOGRAPHY

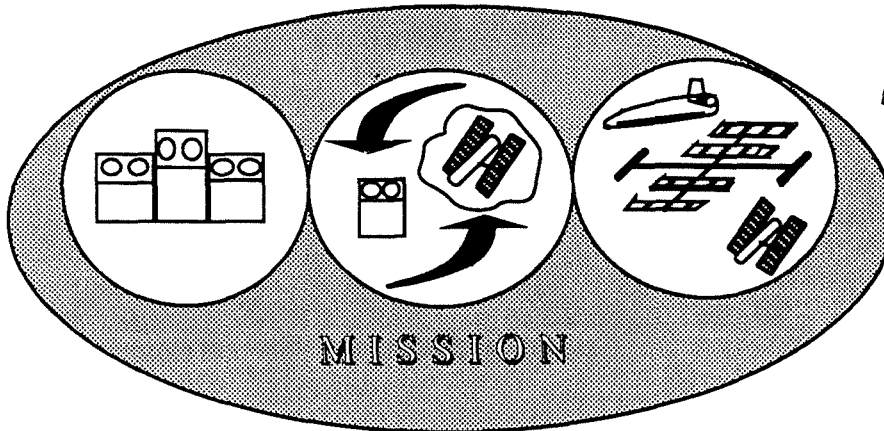
Dr. Alan Burns is jointly employed by the University of York, UK (where he is a Reader in Computer Science) and the Dependable Computer Systems Centre. The latter being a research centre funded by British Aerospace and focused on the issues surrounding the production of safety critical software. Dr. Burns' research is primarily concentrated on real-time systems. He has also been involved in the assessment of proposals for Ada9X. He has published over 120 papers and 6 books, and is involved in a number of projects including the European-wide PDCS (Predictably Dependable Computer Systems) and ESA funded work on design methods and scheduling theory.

MISSION Research

Dr. Colin Atkinson

BIOGRAPHY

Dr. Colin Atkinson, co-PI of the MISSION Project, is an assistant professor of software engineering at UHCL. Prior to this, he was involved in the development of methodologies for the distribution of Ada, and in the design of an object-oriented Ada enhancement known as DRAGOON which facilitates the description of concurrency and distribution in object-oriented systems.



199 510 2274
324052
67

MISsion and Safety CRitical SuppOrt eNvironment

Executive Overview

by:

Dr. Charles McKay

Dr. Colin Atkinson

Motivation and Goals

MISSION is concerned with MASC (Mission And Safety Critical) Systems which are :

- Large
- Complex
- Non-stop
- Distributed
- Real-time

For this kind of MASC system, there is a need to :

- improve definition, evolution and sustenance techniques,
- lower development and maintenance costs,
- support safe, timely and affordable system modifications,
- support fault tolerance and survivability.

The goal of the MISSION project is to :

"lay the foundation for a new generation of integrated systems software providing a unified infrastructure for MASC applications and systems"

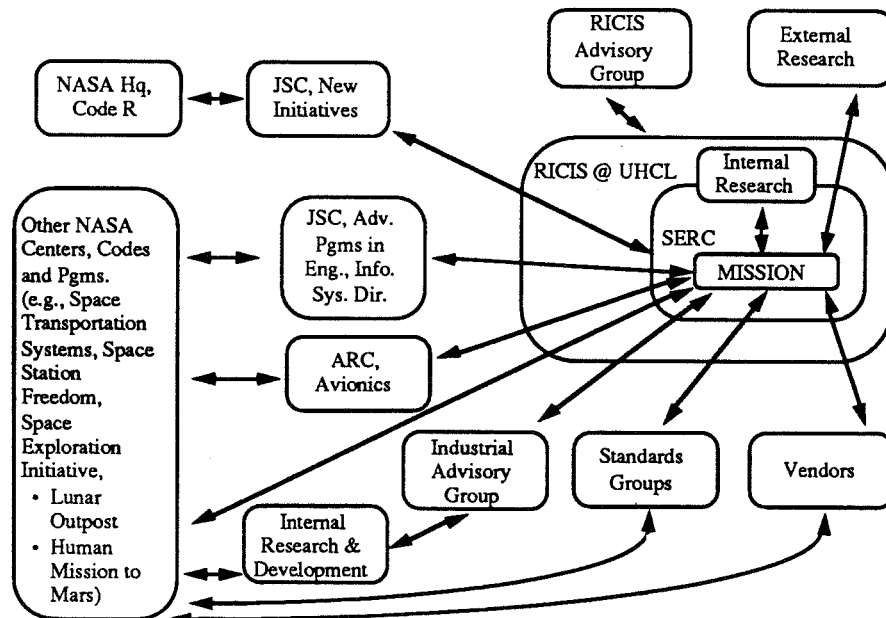
This will involve the definition of :

- a common, modular target architecture.
- a supporting infrastructure.

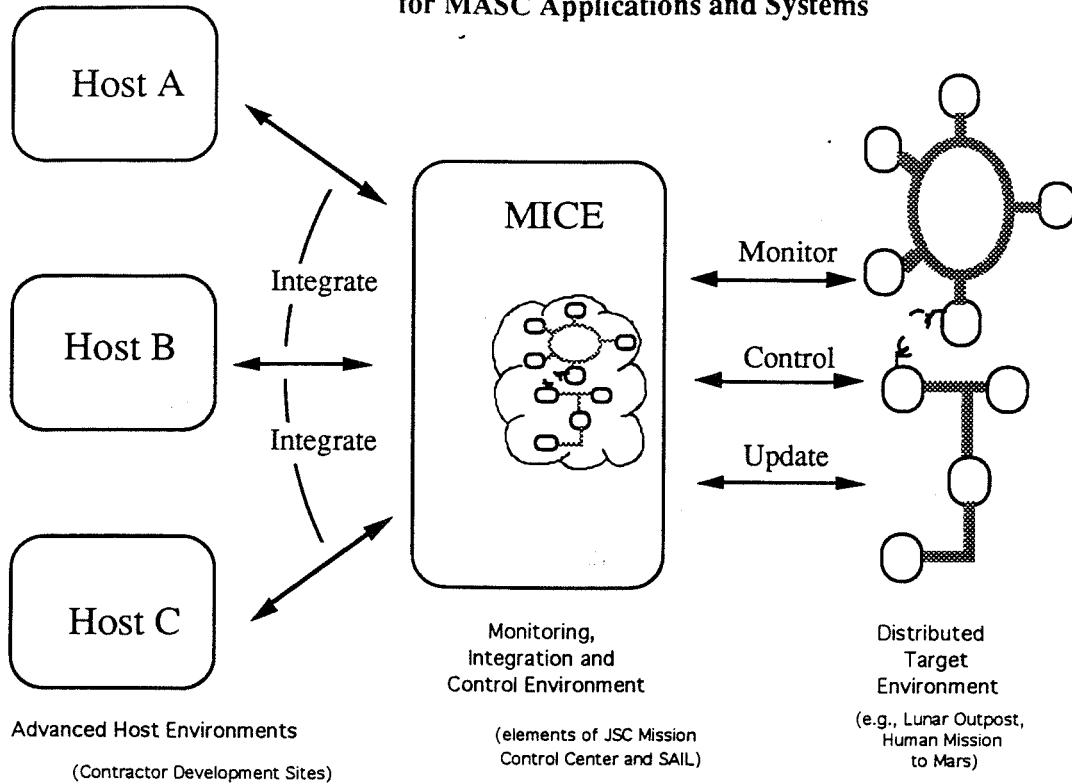
Background

<i>SIZE</i>	21 man years
<i>DURATION</i>	1990 .. 1996
<i>SPONSOR</i>	NASA Headquarters, Code R (through RICIS)
<i>ADVISORS</i>	Industrial Advisory Group (IAG)
<i>Co PI's</i>	Dr. C.W. McKay & Dr. C. Atkinson
<i>PAST CONTRIBUTORS</i>	<ul style="list-style-type: none"> • University of Bradford (Dr. Alan Burns) • Softech • GHG Corporation • Honeywell (Minneapolis) • Softlab (Munich)

MISSION Interaction Diagram



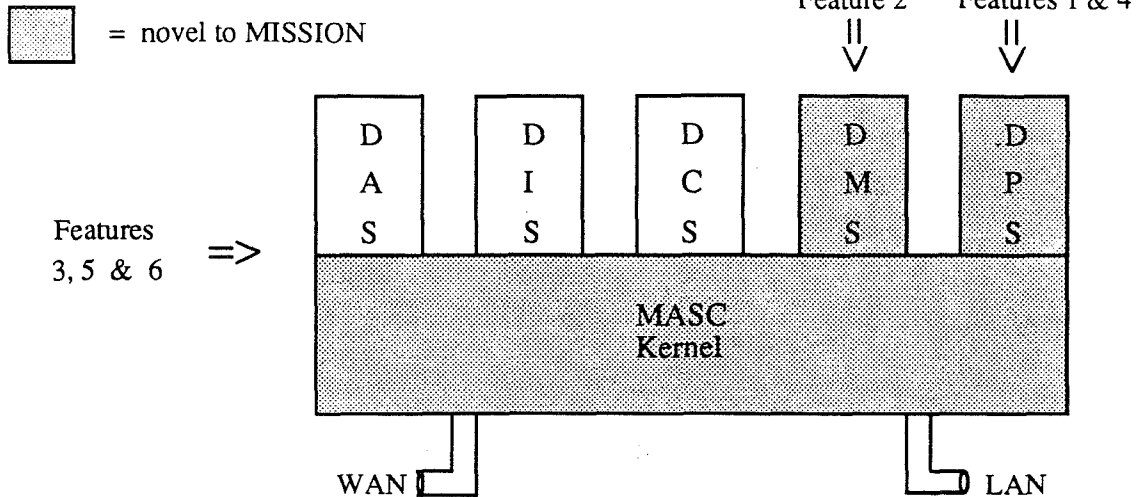
Integrated Life-Cycle Support Environment for MASC Applications and Systems



Requirements versus Features Matrix

	System Goals						Target Features
	Fault-tolerant	Maintainable	Distributed	Survivable	Non-stop	Reliable	
1	•	•	•	•	•	•	On-board software models for monitoring and control
2	•			•			Dedicated software for system level fault tolerance and survivability
3		•			•	•	Separation of policies and mechanisms
4	•	•		•	•		Adaptable run-time policies during non-stop operation
5		•	•		•		Use of a full, concurrent object-oriented, paradigm
6	•			•		•	Firewalling of application and system objects
7						•	Multiple and adjustable levels of security and integrity
8			•				Synchronous and asynchronous communication mechanisms
9	•			•			Distributed nested transactions
10	•		•				Unique identification of all network messages
11	•			•			Redundancy management
12	•			•			Stable storage support for recovery

Generic System Architecture (GSA) for the Distributed Target Environment (DTE)



DAS	Distributed Application System	MASC	Mission And Safety Critical
DIS	Distributed Information System	LAN	Local Area Network
DCS	Distributed Communications System	WAN	Wide Area Network
DMS	Distributed Monitoring System		
DPS	Distributed Policy System		

GSA Requirements on Supporting Infrastructure

Monitoring, Integration and Control Environment (MICE)

- Maintenance of precise models which describe the DTE :-
 - software,
 - hardware ,
 - communications links,
 - human-machine interfaces ,
 - interactions with the environment.
- Distributed Command Interpreter
- Symbolic Diagnostic System

Advanced Host Environment (AHE)

- Construction of precise models of the DTE components
- Rigorous life-cycle approach to evolution and sustenance
- Precise software process models
- Support for special tools and modeling representations.

MISSION's Contribution

Distributed Target Environment

- GSA Requirements,
- GSA Interface Specifications,
- Guidelines for Applying, Tailoring, Modifying and Extending GSA,
- Proof-of-Concept Prototypes of Key and Unique Features.

Monitoring, Integration and Control Environment

- Form of semantic models,
- Guidelines for utilizing semantic models in MICE and DTE,
- Distributed Command Interpreter (DCI) interface.

Advanced Host Environment

- Process Model,
- Model-based life-cycle activities (CLAR/CLAD/CLAIM),
- Prototype semantic model repositories (LMS/OMS).

Anticipated Benefits

Improvements in :

Safety

- fault tolerance
- survivability (availability)
- risk management / certification

Adaptability

- upgrade interoperability
- dynamic reconfiguration

Cost Effectiveness

- reuse
- maintainability
- extensibility

Anticipated Application

NASA Future Programs

- Lunar Outpost
- Manned Mission to Mars

Upgrade to Current NASA Programs

- Space Shuttle
- Space Station

Other MASC Application Areas

- Avionics Systems
- Integrated Weapons Control Systems
- Industrial Process Control
- Transportation Systems
- Hospital Monitoring Systems

Schedule Overview

Significant accomplishments:

- Established MISSION test bed
- Defined semantic modeling representations in Ada-IRDS
- Prototyped Object and Library Management Systems
- Produced distributed nested transactions simulation
- Participated in relevant international standards groups

Future Milestones:

FY93

- Begin second iteration of key components of the GSA
- Specify interface sets for first iteration of GSA study (with simplifying assumptions)

FY94

- Specify interface sets for second iteration of GSA study (without simplifying assumptions)
- Begin second iteration of study of key infrastructure components

FY95/96

- Complete proof-of-concept prototypes of key and unique features of the GSA
- Complete specifications of the key infrastructure components

Session II

**GENERIC ARCHITECTURES FOR
FUTURE FLIGHT SYSTEMS**

3:45 - 5:00

Session Leader:

David Pruett, *NASA/Johnson Space Center*

Presenters:

Generic Architectures for Future Flight Systems

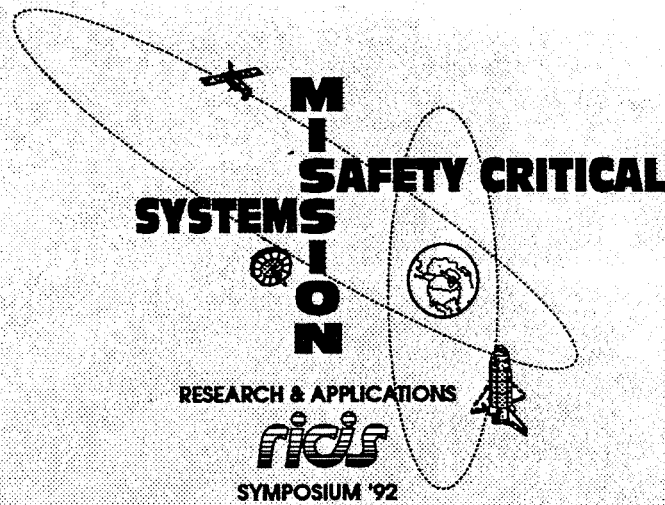
Richard J. Wood, *USAF/Rome Laboratory*

Generic Data System Avionic

Dick Wray, *Lockheed Electronics and Space Company*

**Evolutionary Telemetry & Command Processor (TCP)
Architecture**

John R. Schneider, *Fairchild Space*



GENERIC ARCHITECTURES FOR FUTURE FLIGHT SYSTEMS

SESSION LEADER: *David Pruett*

BIOGRAPHY

Mr. David Pruett joined NASA as a Cooperative Education Student in 1967 and has worked primarily at the Johnson Space Center. He is currently the Manager for Advanced Programs in the Flight Data Systems Division. Prior to that he had been involved with the Space Station Program in various positions including Chief, Systems Development Branch where he was responsible for the Space Station Freedom Data Management System (DMS) System Software, the Work Package 2 Avionics Development Facility and Integration Test and Verification Environment and Chief, Space Station Information Systems, Engineering Branch, Space Station Program Office, Reston. His current interest is in the development of standards based architectures for flight data systems and avionics in general. He holds a B.S. in Mathematics earned in 1970 from Texas A&I University.

27233

GENERIC ARCHITECTURES FOR FUTURE FLIGHT SYSTEMS p. 16**RICHARD J. WOOD
USAF/ROME LABORATORY**

1995107745

527053

16p.

ABSTRACT

Generic architectures for future flight systems must be based upon Open System Architectures (OSA). This provides the developer and integrator the flexibility to optimize the hardware and software systems to match diverse and unique applications requirements. When developed properly OSA provides interoperability, commonality, graceful upgradeability, survivability and hardware/software transportability to greatly minimize Life Cycle Costs and supportability. Architecture flexibility can be achieved to take advantage of commercial developments by basing these developments on vendor-neutral commercially accepted standards and protocols. Rome Laboratory presently has a program that addresses requirements for OSA as will be presented.

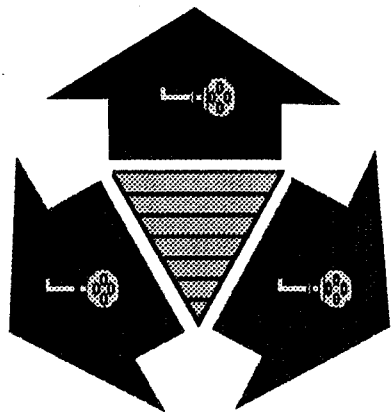
BIOGRAPHY

Mr. Richard J. Wood is the Laboratory Program Manager for the Architecture for Survivable Systems Processing (ASSP) program. He has managed several R & D programs such as Advanced On-Board Signal Processors and Passive Tactical Target Identification Development. Prior work includes development of automatic navigation and landing systems for the Naval Aviation Facilities Experimental Center and the Federal Aviation Agency.

GENERIC ARCHITECTURE FOR FUTURE FLIGHT SYSTEMS

THE

ASSP



RICHARD WOOD
ROME LAB/OCTS
GRIFFISS AFB NY 13441-5700
COMMERCIAL (315) 330-2215
DSN 587-2215

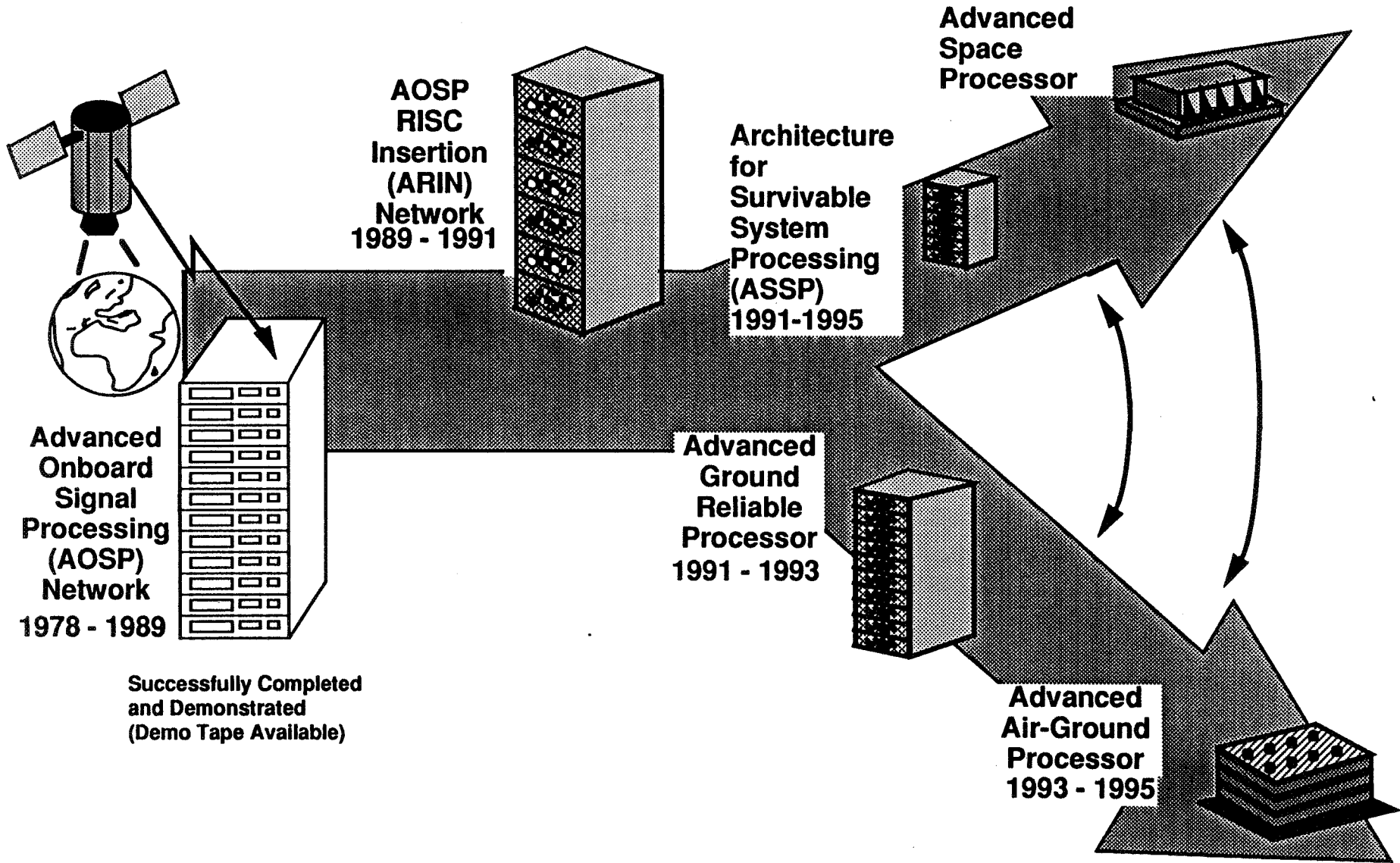
INTRODUCTION

- **Background**
- **Issues**
- **ASSP Profile**
- **Products/Benefits**
- **Synergism**
- **Summary**

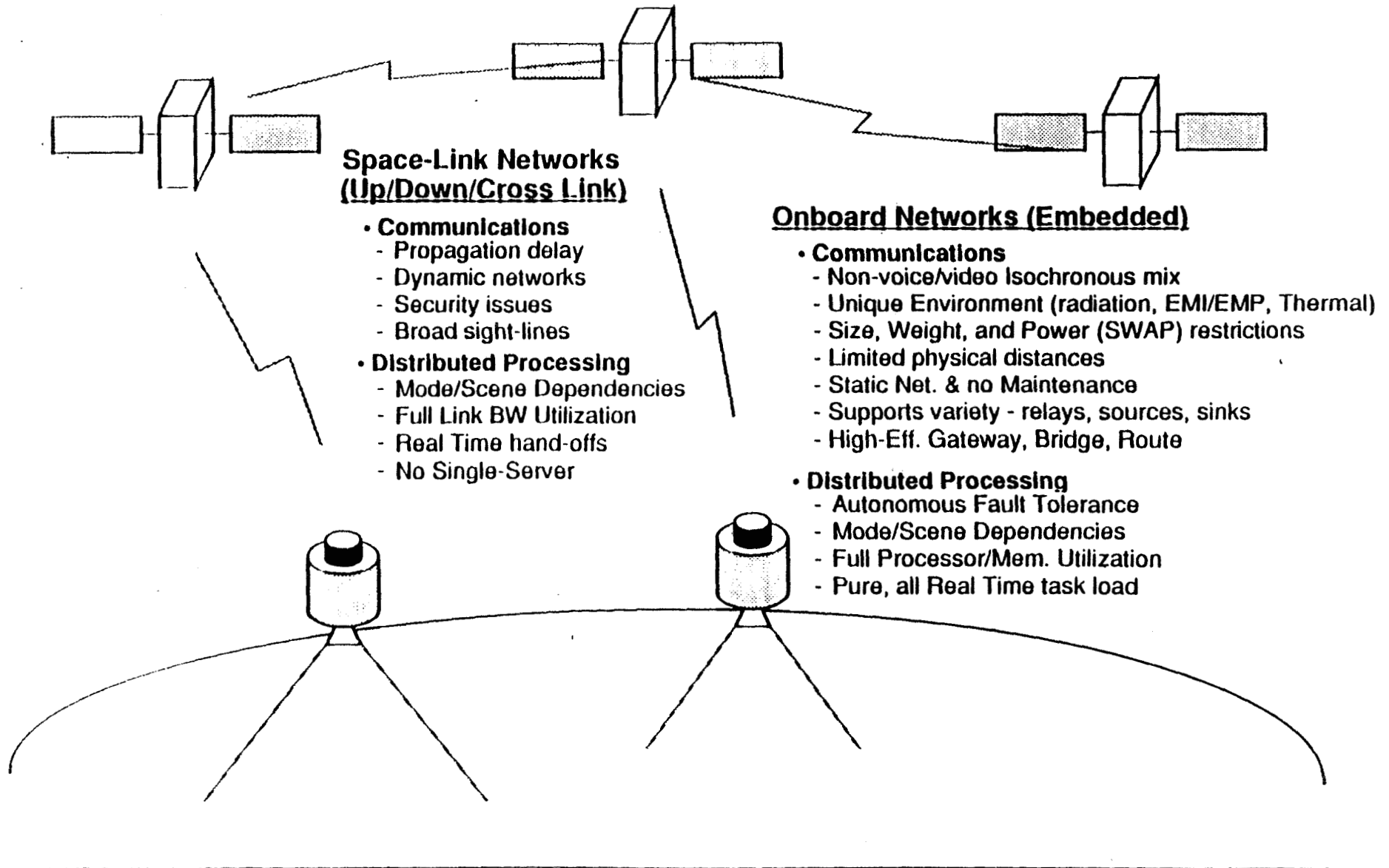
NUMEROUS PROGRAMS ADDRESSING PORTIONS OF OSA/OSI

- AF / RL - Architecture for Survivable System Processing (ASSP)**
- Corporate Information Management (CIM)**
- Modular Open System Architecture Standard (MOSAS)**
- Consultive Committee for Space Data Systems (CCSDS)**
- Common Communication Components (Com³)**
- Next Generation Computer Resources (NGCR) Program**
- NORAD - US Space Command Integrated Command and Control System (NUICCS)**

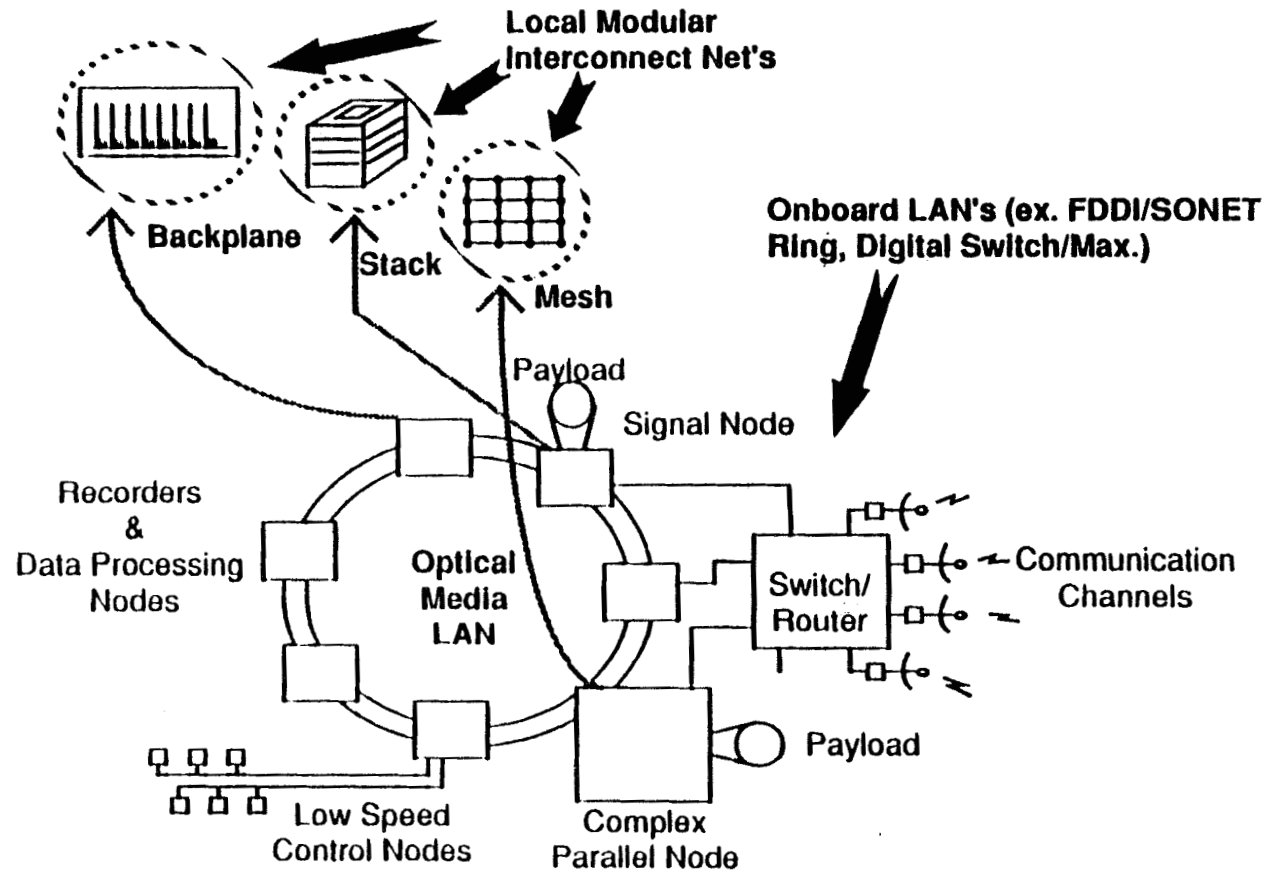
SURVIVABLE SYSTEM PROCESSING TECHNOLOGY OPPORTUNITY



Unique Space and Onboard Issues



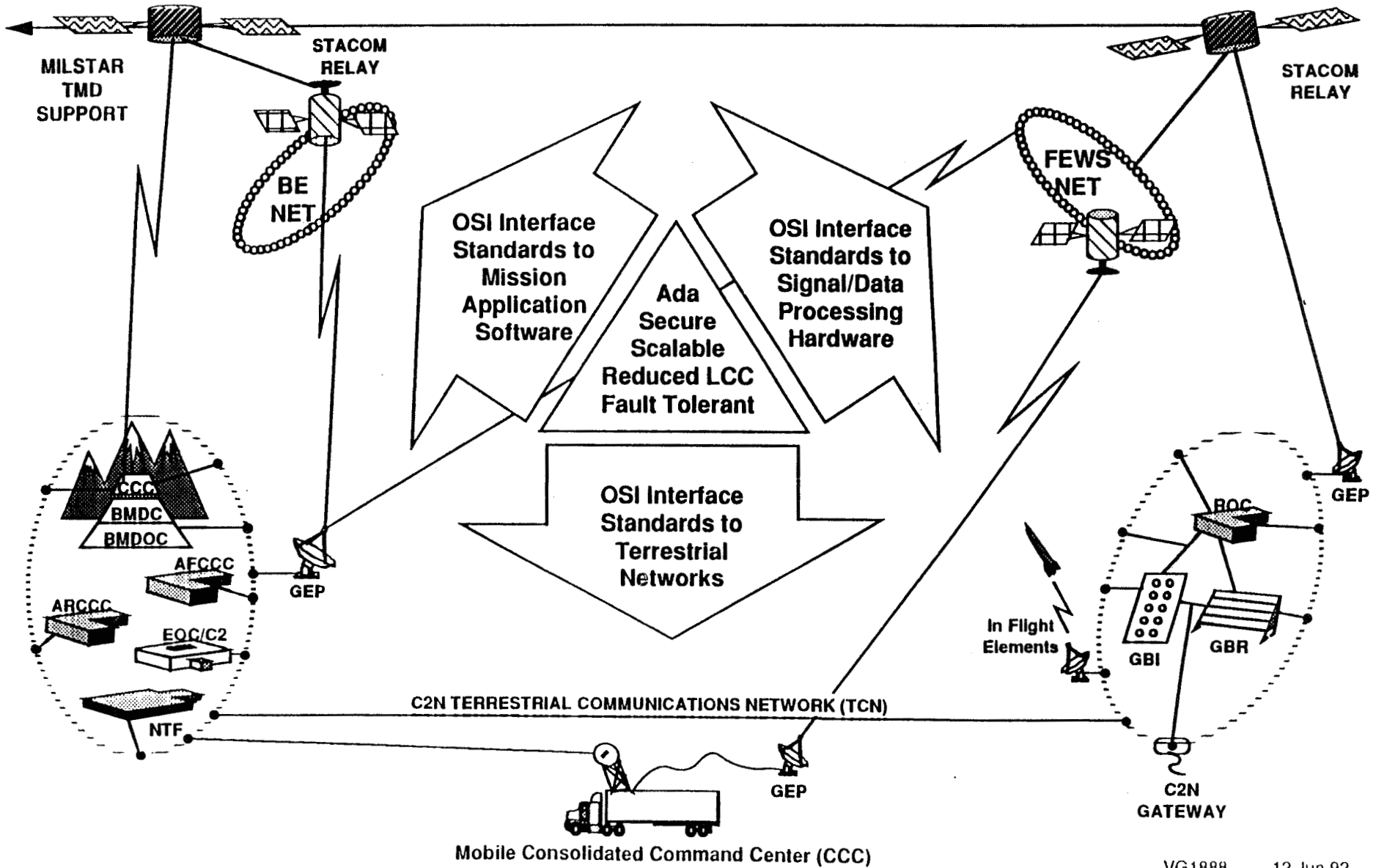
Complex Onboard Networking Scenario



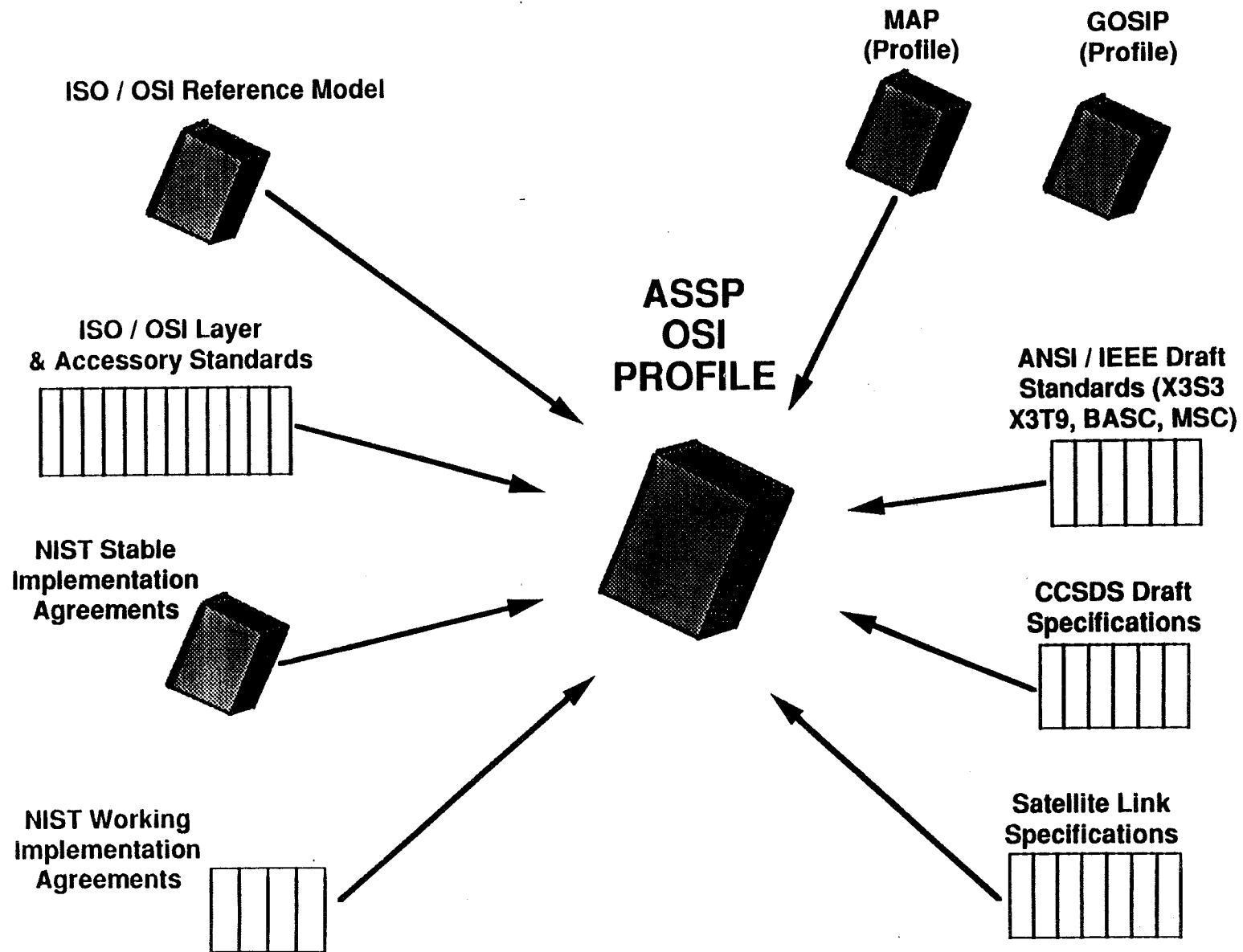
Future satellites will incorporate a wide variety of functionality implying the need for varied processors and subnetworks.

ARCHITECTURE FOR SURVIVABLE SYSTEM PROCESSING (ASSP)

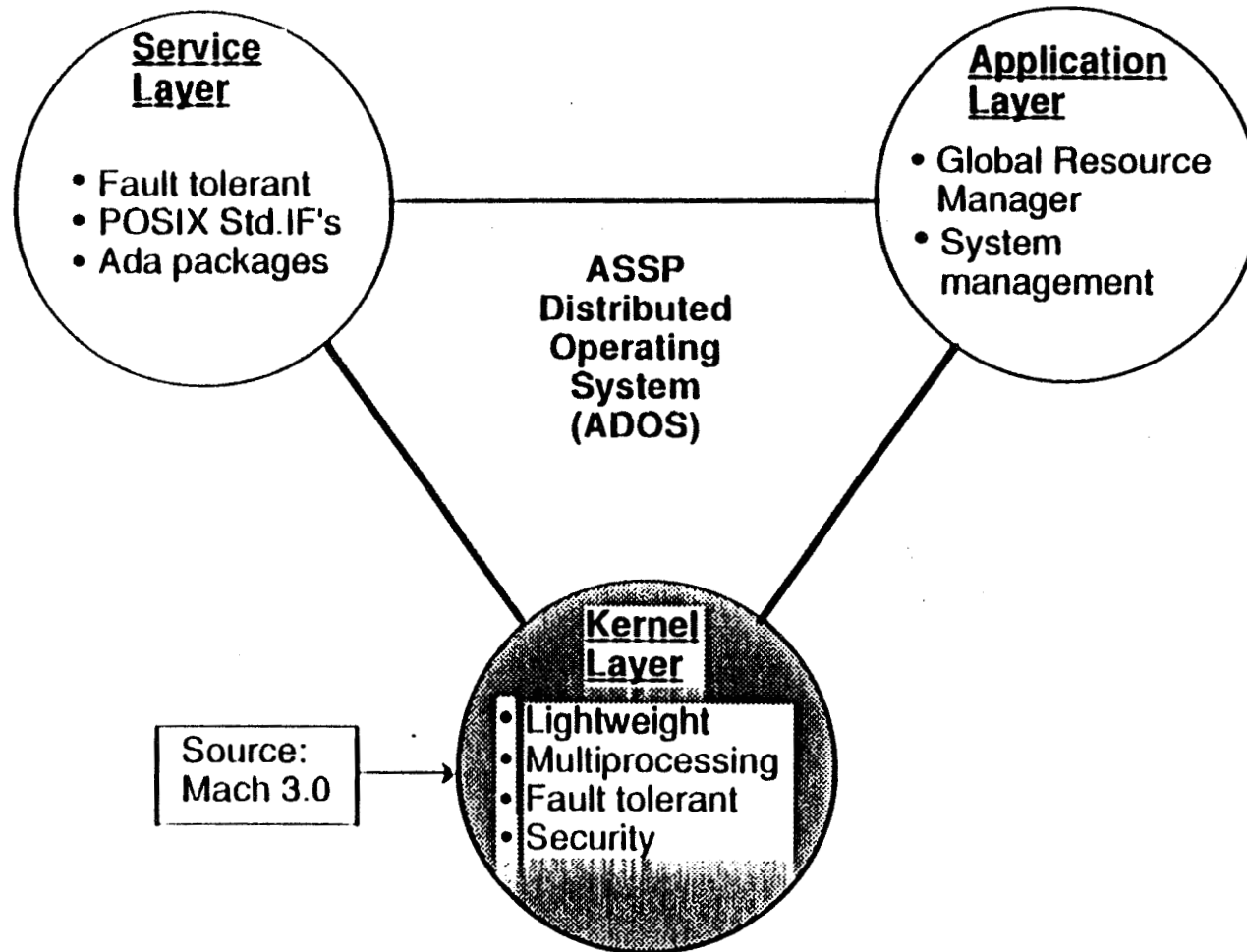
Objective: Develop Standards Based OSA to Ensure Interoperability, Commonality Among Processor Components

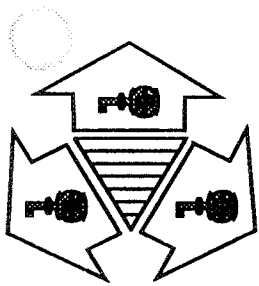


ASSP OSI PROFILE CREATION



Baseline Kernel

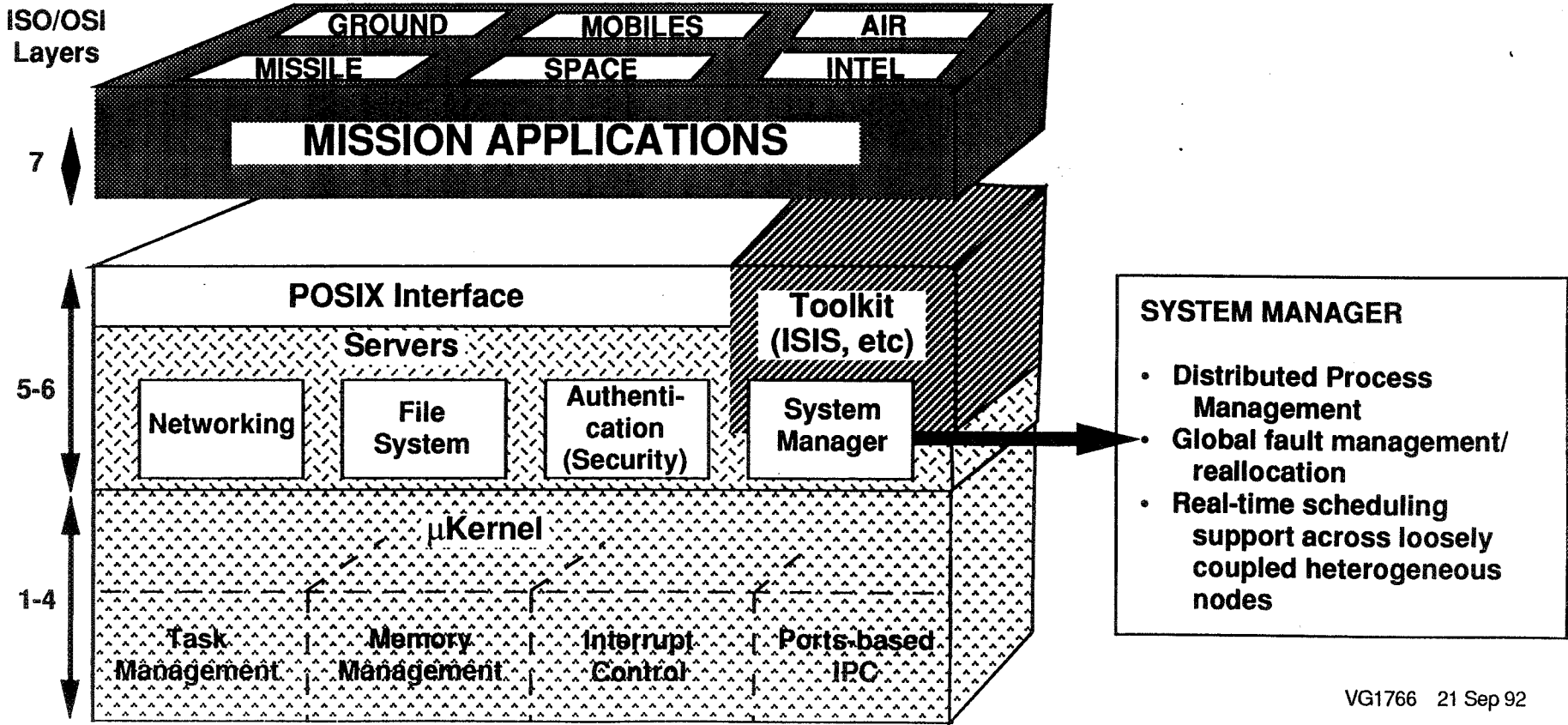




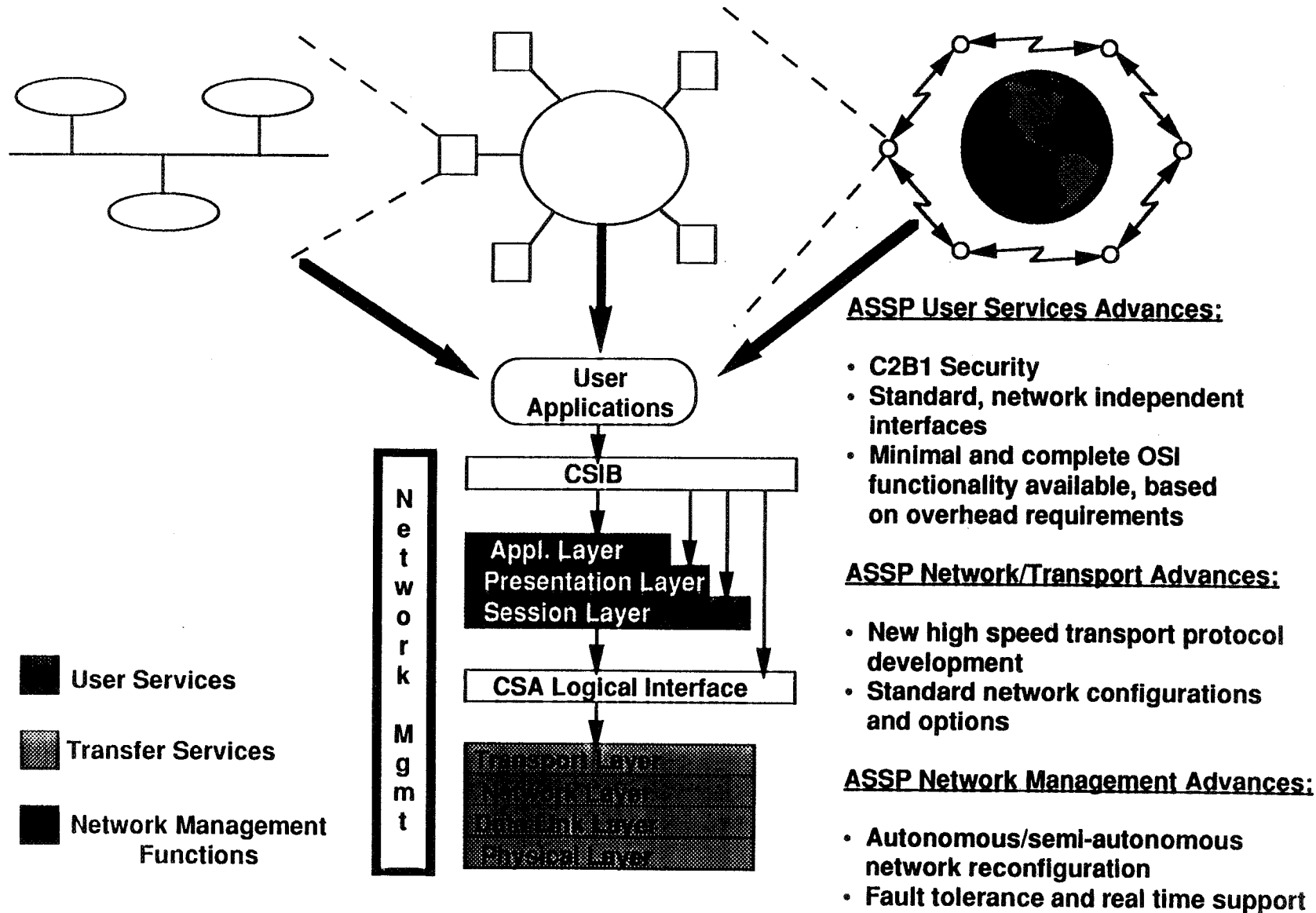
ASSP DISTRIBUTED OPERATING SYSTEM (ADOS)

- Layered ADOS Architecture provides optimal mixture of features vs. efficiency
 - Microkernel provides efficient, real time, distributed functionality
 - Services provide rich networking, security, and file system management

- ASSP-specific requirements implemented using application and system manager functions when possible



NEXT GENERATION OSI PERFORMANCE



ASSP will take advantage of discrete advancements in protocol, service, and management to provide verified open specifications that facilitate interoperability and next-generation performance.

ASSP PRODUCTS

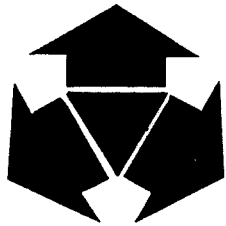
- **Survivable Systems OSI Architecture Specification**
 - **Profiles composed of standards**
 - **Options and enhancements tailored to space systems**

 - **Flexible/Adaptable Distributed, Real Time Operating System**
 - **Modular and tailorable**
 - **Fault tolerance support**
 - **OSI communication support**
 - **Standard Interface**

 - **Simulations**
 - **Support for specific system/application designs**
 - **Users' manual**
 - **Model libraries**

 - **Breadboard and Advanced Technology Testbeds**
 - **Configuration platform for realtime testing, prototyping, and conformance testing**

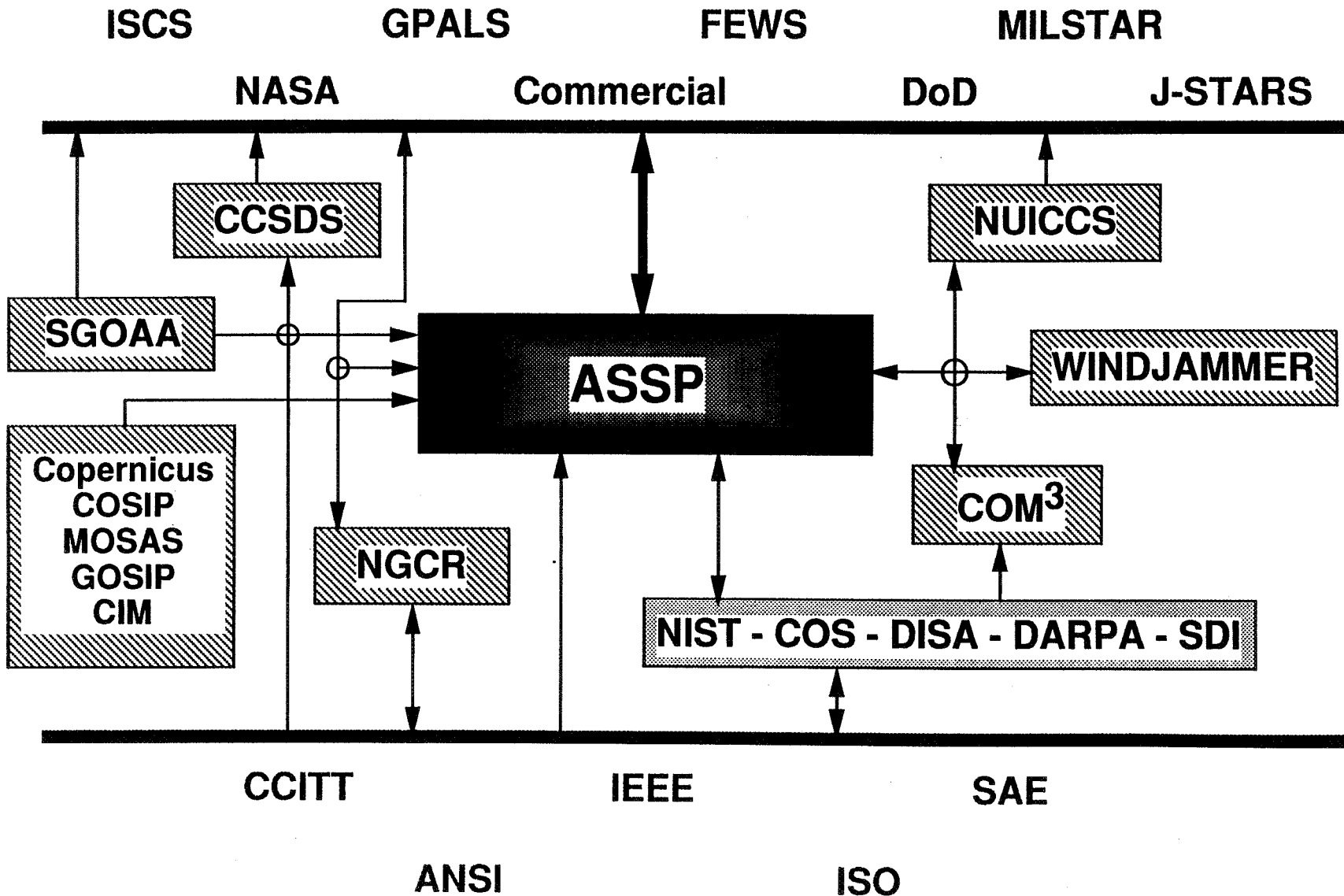
 - **Hardware Specifications and VHDL Designs**
 - **Space-qualifiable networking components**
-

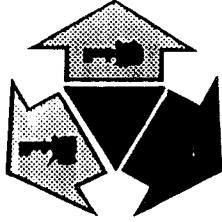


ASSP STANDARDIZATION BENEFITS

- * **INTEROPERABILITY** (Commonality) → Satisfies data/communication message passing requirements between intra-platform components and ground assets as well as inter-platform (via GPALS standards).
Provides interoperability with other system constellations FEWS, GPS, BP, MILSTAR, BE, GBR, GBI
- * **SUPPORTABILITY** → Reduces system Life Cycle Cost (LCC)
→ Enhances logistics requirements for maintainability and readiness.
- * **UPGRADEABILITY** → Reduces machine dependency, promotes portability, enhances software reuse, eliminates conforming to obsolete systems.
- * **LONGEVITY** → Provides means for new generation components to incrementally upgrade older but still operational systems with no impact on operational status
- * **AFFORDABILITY** → Reduces LCC.
→ Eliminates sole source costs.
→ Eliminates system obsolescence.
→ Maintains system effectiveness throughout long term requirements.
→ Leverages on commercial developments

ASSP SYNERGISM





ASSP SUMMARY

THE ARCHITECTURE FOR SURVIVABLE SYSTEM PROCESSING PROGRAM ADDRESSES THE KEY TECHNICAL CHALLENGES OF:

- **INTEROPERABILITY/INTERCHANGEABILITY OF HETEROGENEOUS PROCESSING NODES**
- **OPEN SYSTEMS ARCHITECTURES FOR SPACE, AVIONICS AND GROUND ALLOWING RAPID INSERTION OF NEW MILITARY AND COMMERCIAL TECHNOLOGY**
- **GOSIP COMPATIBLE / COMPLIANT**
- **ISO / OSI REFERENCE MODEL BASELINE**

N95-14160

27234

27234

P. 12

1995107746

324055

ARCHITECTURE FOR SURVIVABLE

SYSTEMS PROCESSING

(ASSP)

Technology Benefits for

OPEN SYSTEM INTERCONNECTS

24 JULY 1992

RICHARD J. WOOD

RL/OCTS

ASSP Program Manager

TABLE OF CONTENTS

1.0	INTRODUCTION	1
2.0	PROBLEM STATEMENT	2
3.0	ASSP NETWORKING SOLUTIONS.....	3
3.1	Application Through Network Layers.....	5
3.2	Link Layer and Physical Layer.....	6
4.0	OPERATING SYSTEM SOLUTIONS.....	8
5.0	SOFTWARE DEVELOPMENT ENVIRONMENT.....	9
6.0	ASSP SYSTEM SIMULATION SOLUTIONS.....	10
7.0	CONCLUSION	10

FIGURES

FIGURE 1-1:	Application of Networks within a Surveillance Satellite.....	1
FIGURE 2-1:	ASSP User Community	3
FIGURE 3-1:	ISO/OSI Reference Model.....	4
FIGURE 3.1-1:	ASSP OSI Architecture.....	5
FIGURE 4-1:	ASSP Operating System Diagram.....	9

1.0 INTRODUCTION

The Architecture for Survivable Systems Processing (ASSP) program is a two phase program whose objective is the derivation, specification, development and validation of an open system architecture capable of supporting advanced processing needs of space, ground, and launch vehicle applications.

The output of the first phase is a set of hardware and software standards and specifications defining this architecture at three distinct levels. The lowest level consists of the hardware bus(es), operating system, and software development environment. It facilitates the interoperability/interchangeability of heterogeneous processors for the processing subsystem. The middle level is the intraplatform networking of the subsystems such as the data processing subsystem with the communication and other subsystems, as shown in Figure 1-1. The sensor and signal processing subsystems, although frequently connected by point-to-point links, may also be candidates for networking technology. The top level is the interplatform networking between common platforms, and to platforms of other system elements. In most cases, existing standards will be adequate; specification of these standards is all that will be required. Where standards do not exist or are inadequate, specifications will be developed. These specifications will be approved by the government Technical Advisory Group (TAG) to insure non-parochial interests and eventual acceptance by appropriate standards committees. The first phase output will include breadboard implementations of ASSP specified interconnection technology. This breadboard will be implemented in flight deployable hardware, to demonstrate the architectures capabilities as defined in the SOW.

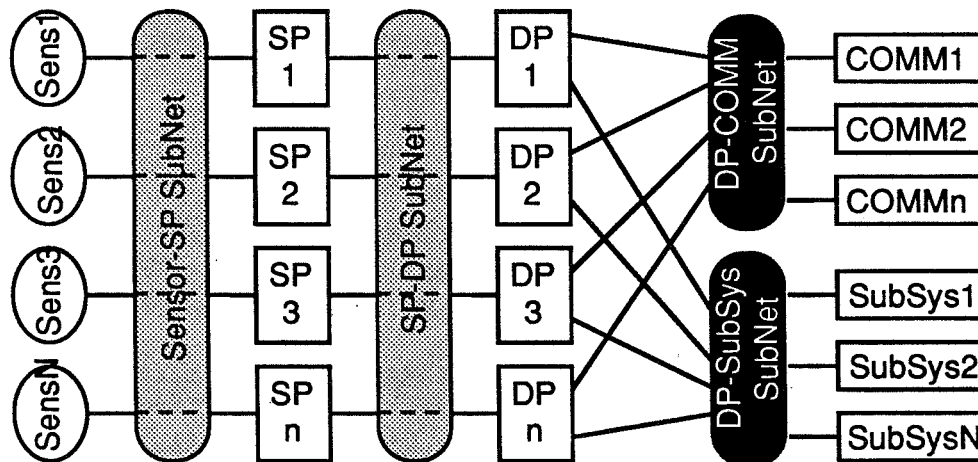


FIGURE 1-1: Application of Networks within a Surveillance Satellite

The second phase of the ASSP program will validate these standards and develop any technology necessary to achieve strategic hardness, packaging density, throughput requirements, and inter-operability/inter-changeability.

2.0 PROBLEM STATEMENT

The Follow - on Early Warning System (FEWS) and Brilliant Eyes (BE) are some of several specific programs whose requirements are being used to drive the ASSP program. They were selected because of stressing operational requirements, that will need:

1. **Interplatform communication standards** between satellites from different contractors and ground stations.
2. **Backplane and intrasatellite network standards** to minimize breakage during Block to Block transitions as the processing loads shifts from ground to space.
3. **Interplatform communication standards** between Satellite constellations (BE, FEWS, DSP-upgrades etc. terminal defense systems (GSTS, THAAD, E²I, etc.), and the ground stations during EOAC and GPALS. Figure 2-1 summarizes the benefits of ASSP in this context.

The selection of standard intrasatellite network architectures can minimize the re-design of the interfaces between subsystems during block upgrades, Similarly the selection of a standard backplane can simplify upgrading from Block to Block. For example, a 16-bit processor can be replaced with a 32-bit processor that both interface to the same backplane, yielding an order of magnitude improvement in throughput with minimal impact on other board designs. Finally, the selection of a standard operating system interface will allow applications software developed to be reused ensuring cost effective transitions.

The selection of interplatform communication standards mitigates the risk that Satellite constellations can interchange handoff messages with interceptor systems, and deliver summary messages direct-to-users.

3.0 ASSP NETWORKING SOLUTIONS

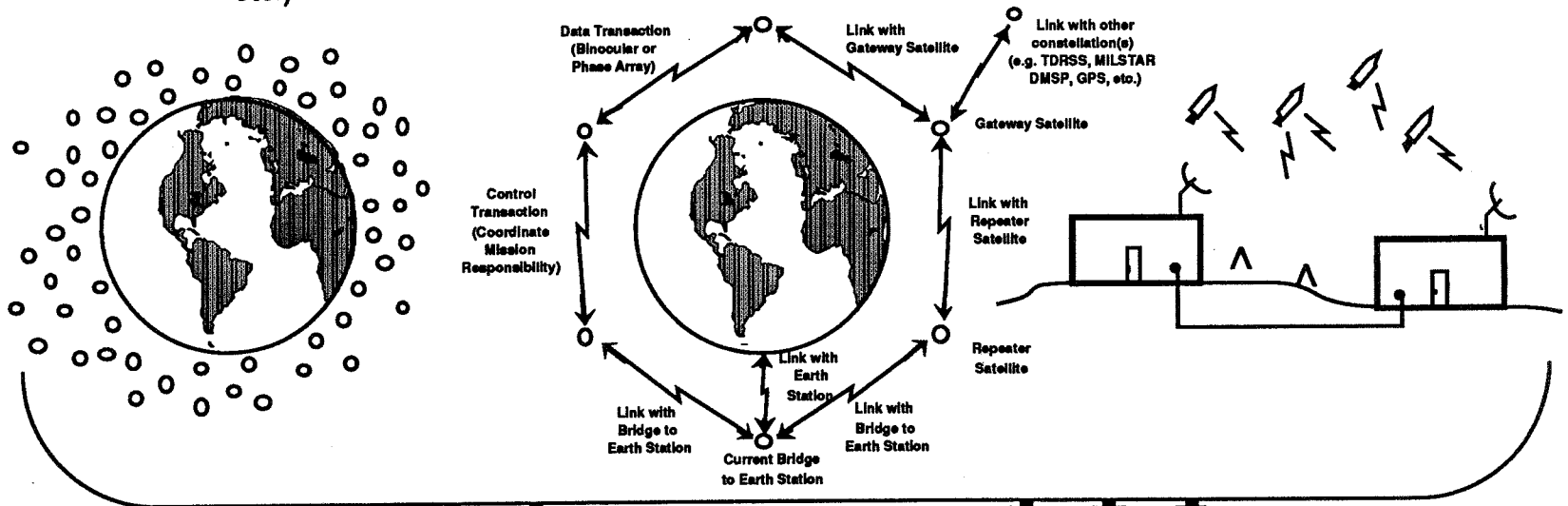
The International Standards Organization (ISO) developed the Open Systems Interconnect (OSI) standard in response to the growing need in the commercial community for interoperability among different computer vendors. That same need is now manifesting itself in the defense community. ASSP was conceived with the idea of taking advantage of the significant commercial investment in OSI technology, while also bridging the differences in networking requirements between the commercial and defense communities. BE and FEWS are examples of programs that have portability, upgradeability, and interoperability requirements that can be met through the work being performed on the ASSP program. This section will provide a brief overview of the state-of-the-art commercial OSI technology and how ASSP is utilizing that to solve problems for programs like Integrated Satellite Control Systems (ISCS), Common Communications Components (COM³), and Corporate Information Management (CIM)

**PROLIFERATED
CONSTELLATION
SPACE SYSTEMS**
(Brilliant Eyes, Brilliant Pebbles,
etc.)

**LARGE
PLATFORM
SPACE SYSTEMS**
(FEWS, DSP, Milstar, SDI
EXP'S, NASA/NOAA)

**TERMINAL
DEFENSE
SYSTEMS**
(GSTS, Probes, Interceptors)

FIGURE 2-1: ASSP User Community



ASSP Technology Benefits for OSI

- INTEROPERABILITY
- RELIABILITY
- AVAILABILITY
- FAULT TOLERANCE

- EXPANDABILITY
- EFFICIENCY (SWAP)
- FLEXIBILITY
- SECURITY

Figure 3-1 shows the layers that make up the OSI Reference Model. This model is based upon the well-known principle of "information hiding". To reduce design complexity and promote portability and interoperability the OSI Reference Model uses layers, where each layer builds upon the services/functions offered by its predecessor. The purpose of each layer is to offer specific services to the higher layers, shielding those layers from the implementation details of how the service is implemented. Thus, by standardizing only the interface between layers, and not the implementation, implementations can change without having to modify higher layers. For example, the Network Layer is primarily concerned with routing messages between processors. If the routing scheme were to change, the Transport Layer and all higher layers are shielded from this change - requiring little or no modification to their implementation. This is a powerful concept that allows much easier insertion of new communication technologies with minimal cost and schedule impact.

The challenge facing the military community is how to take advantage of this networking standard and the significant commercial investment in associated technologies. Most of the commercial use of the OSI standard is in Local Area Networks (LANs) on the manufacturing floor or in the office. Although the defense community can make use of this work (and is), the real need is for the use of standards in embedded systems. The embedded environment offers much more stringent communication requirements than the corresponding manufacturing or office environments. In addition, high throughput density and reliability requirements, and the emergence of advanced backplanes, combine to cause networking concepts to be applied at the backplane as well as the traditional LAN level. Very high speed, efficient communications are required in an environment that places severe restrictions on size, weight, and power. Thus, communication protocols must be specified/developed that are not computer and/or memory intensive. ASSP's mission is to provide a suite of protocols that satisfy the OSI standard and meet the requirements of the embedded environment.

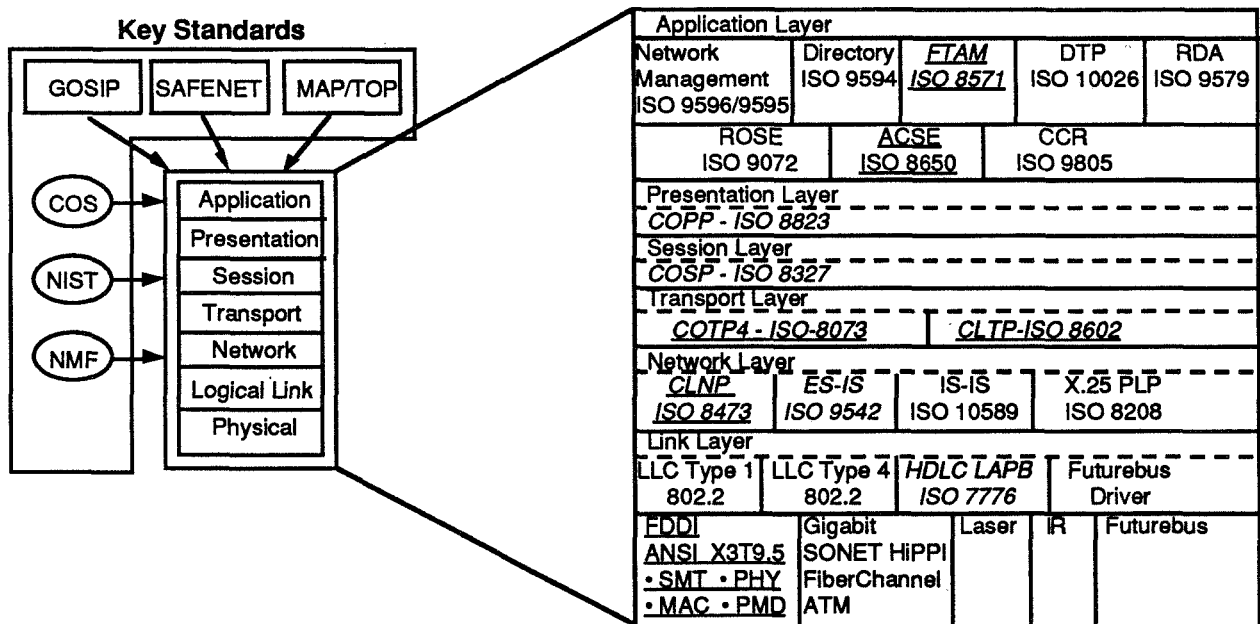
Layer No.	Name	Protocol
7	Application	Supplies user services, application-specific standards
6	Presentation	Provides code conversion, data reformatting
5	Session	Coordinates interactions between end application processes
4	Transport	Ensures end-to-end data integrity and quality of service
3	Network	Switches and routes information
2	Data Link	Transfers information to other end of physical channel; controls media access
1	Physical	Transmits bit stream over the medium

FIGURE 3-1: ISO/OSI Reference Model

3.1 Application Through Network Layers

The Application Layer through-the Network Layer provide communication services accessible by application programs. These layers provide services such as guaranteed message delivery, checkpoint/rollback, security, encryption, data conversion, etc. The lower two layers (Data Link and Physical) are primarily concerned with managing the physical communication medium (twisted pair copper wire, coaxial cable, fiber optics, parallel backplanes, RF links, etc.) and are discussed separately in Section 3.2.

For each of the layers, the OSI Reference Model specifies numerous options. It recognizes that there is not one all encompassing set of network functionality that satisfies the communication needs of every user. Thus, for each of the OSI layers, there exist many protocols that provide all or part of the functionality specified by the OSI Reference Model. In addition, there exist standard "profiles" which specify a suite of protocols that provide a single OSI implementation. For example, the MAP and TOP profiles specify a standard OSI implementation (including specific options that are mandatory) for manufacturing and office environments. Each specifies protocols that provide functionality that best satisfies the needs of each environment. ASSP will develop a similar embedded systems OSI profile, where specific functionality is required for each layer in order to interoperate with other systems implementing the same profile. Existing commercial protocols will be used extensively in the creation of this embedded systems OSI profile. Figure 3.1-1 shows some of the sources we will draw upon as well as a tentative list of specific protocols that will make up the embedded systems OSI profile.



- Items in *italics* are part of the GOSIP standard
- Items underlined are part of the SAFENET-II standard

FIGURE 3.1-1: ASSP OSI Architecture

ASSP recognizes that not all applications need to start at the top of the OSI "stack" in order to send/receive messages. Therefore, options of implementing "short stacks" and/or "layer bypassing" will be provided. A "short stack" is one in which only the layers necessary to meet a specific application's communication requirements are present. Thus, the overhead associated with sending/receiving a message can be significantly reduced. The cost for this, of course, is in decreased reliability, functionality, and/or security for the transmitted/received message. However, in embedded systems, this is often an acceptable trade in order to increase communications throughput. "Layer bypassing" can be used to decrease the overhead associated with the transmission/reception of an individual message. With a full ASSP OSI stack present, an application can choose to send a message using the functions associated with specific layer(s). This is useful when only selected messages require faster transmission/reception, while others require the full functionality offered by the ASSP embedded systems profile.

Use of the embedded systems profile developed by ASSP will allow Platforms to meet portability, upgradeability, and interoperability requirements. Its use will allow satellites developed by different contractors to communicate with each other and common ground stations. The upper layers will provide standard functionality that will support this interoperation. Factors such as buffer overflow, message acknowledgement, lost packets, data conversion, etc., that previously fell into the O/S or applications' domain, are handled by the upper layers of the embedded systems profile. Many of the details of the communication links are hidden from application developers, thus each contractor has less complex (and therefore lower risk) software designs. An additional benefit is that specific satellites should be able to easily communicate with other systems (such as BE, FEWS, MILSATCOM, GBR, J-STARS) that also use the ASSP embedded systems profile.

In addition, use of the ASSP embedded systems profile will ease the portability/upgradeability problems like the BE contractor will face in moving from the Block 1 satellites to Block 2. Because applications are hidden from the details of communications and networking, new technologies can be easily inserted without affecting the applications. Only the implementation of the OSI layer(s) that is affected by such a technology insertion would change. Therefore, the risk of upgrading from Block 1 to Block 2 is significantly decreased by the use of the embedded systems profile developed by ASSP.

3.2 Link Layer and Physical Layer

The link and physical layers of data communications within satellite architectures could consist of the hardware required to implement interconnection within and between spacecraft, as well as of up/down link communications hardware. ASSP technology development will result in lower cost, shorter schedules, and improved performance. These benefits accrue from early definition, specification, and development of the principal hardware interfaces required for candidate designs. Four types of interface standards will be developed by ASSP which are projected for Spacecraft application: backplane, high-speed serial, R-F link, and sensor/signal processor interface. Application of these technologies are as follows: backplanes for Payload Data &/or Mission Data Processors; high-speed serial for Communications Subsystem, S/C Control Subnetworks, and S/C Management Subnetworks; radio-frequency for Cross-link and Up/Down link networks; and, sensor/signal processor

interfaces for interface between Sensors and Signal Processors, or between Signal Processors and Data or Mission Processors.

Interconnection standards specification and technology development independent from system contractors is appropriate because an independent effort, such as ASSP, can address the needs of all system elements and ensure compatibility between BE, FEWS, and other GPALS elements. In addition, the independent specifications provided by ASSP will facilitate technology insertion and lower risk for individual projects.

First, ASSP will develop not just technology trades and roadmaps, but detailed specifications for a backplane, high-speed serial network, R-F network, and sensor/SP I/F's which are specific to space-based surveillance platforms. These specifications will include the following:

- **Physical Interfaces** - drivers/receivers, connectors, media characteristics, frequency and loading performance, etc.
- **Media Access Protocol** - encoding, clocking, arbitration, control/modes, handshake, error detection, etc.
- **Logical Link Services** - virtual channels, addressing, flow control, buffer management, burst and latency performance, multiprocessing and DMA support, error recovery, etc.

These specifications will constrain implementation of the interconnection hardware for the four target technology types (backplane, high-speed serial, etc.). The advantages of ASSP specifications will be that they will make maximum use of advanced "open", nonproprietary standards such as Futurebus+ and FDDI, and will extend them to accommodate size, weight, power, environment (including radiation), fault tolerance, and other requirements for space applications. This will allow maximum flexibility in prototyping and software development while providing highly capable technologies for space use.

Second, ASSP will develop components which fill technology gaps, and integrate these with other available components to build and validate integrated interconnection hardware for each of the four interconnection types. This includes hardware design instantiations of the backplane, high-speed serial network, R-F network, and sensor/SP interface specifications. Any new hardware designs will be done in VHDL and targetable to radiation-hardened VLSI libraries. Hardware for each interconnection type will be integrated and will include low-level communications management, buffering, control, protocol, error recovery, encoding, driving/receiving, connectors, and media, among other functions. Verification/validation of each interconnection network type will include demonstration of specific GPALS, FEWS threat detection and tracking applications. This will ensure the ability to handle the desired surveillance communication traffic at all interconnection levels.

Finally, in Phase II ASSP will update protocols and services, provide additional packaging density, and integrate strategically hardened prototype interconnections networks. These upgraded networks will be compatible with prior designs due to the open system development strategy of ASSP. Thus, transition to more stressing

onboard processing for FEWS would be smoothly accommodated using ASSP technology and specifications.

4.0 OPERATING SYSTEM SOLUTIONS

Satellites developed and launched within the Integrated Satellite Control System (ISCS) directives will require the latest software technology to produce realtime, embedded software systems that will ensure performance, reliability and long operational life that will live up to the expectations of the proposed systems.

The ASSP program is the source for the requirements and specifications for such systems. By adopting an "open system" approach and striving to adapt from the best commercial, university, and government non-proprietary products, ASSP will specify an operating system based upon commercial standards, protocols and rules and DoD, NASA are in a position to profit.

Three major structural characteristics point up the utility of the ASSP Operating System (O/S): a standardized application programming interface (POSIX, ISO 9945-1 and ISO 9945-2), support for Ada applications executing in realtime, and portability of the O/S kernel that will ease the risk of hardware upgrade to higher performance processors. The following discusses each of these characteristics in detail.

The IEEE standards group has defined a Portable Operating System Interface, extended (POSIX) for the C language, that initially came out in 1988. It has been extended to cover realtime, multitasking, and the Ada language. This standard defines the function calls to the underlying operating system such that an application will get the same service on any host as long as the host O/S meets the same POSIX standard. The ASSP program is working with the IEEE standards groups to select the proper subset of these specifications that support the BE, FEWS class of problems. By writing the software following the POSIX standard, future upgrades will have minimum software risk.

The Department of Defense has defined Ada (DoD-STD-1815A) to be the high order software language for realtime, embedded systems. This is a large step along the way to having a repository of reusable realtime code. However, an additional step is needed; the application program interface to the operating system on the host computer must also be defined by an accepted standard. Otherwise, the reusable software is the oft-stated 95% complete with the other 5% costing the usual non-recurring.

The ASSP O/S has a layered structure (see Figure 4-1); the kernel layer touches and controls the host hardware, the service/policy layer is made up of the major functions (written in Ada packages) that are needed by the applications, and the applications layer has network and system management software in addition to the mission software written by the user. The service layer can be tailored to the needs of the applications, ensuring minimum data latency and the realtime performance required. The OSI stack for network communications is in the service layer and this standard set of protocols provides for interoperability among satellites in or out of their constellations.

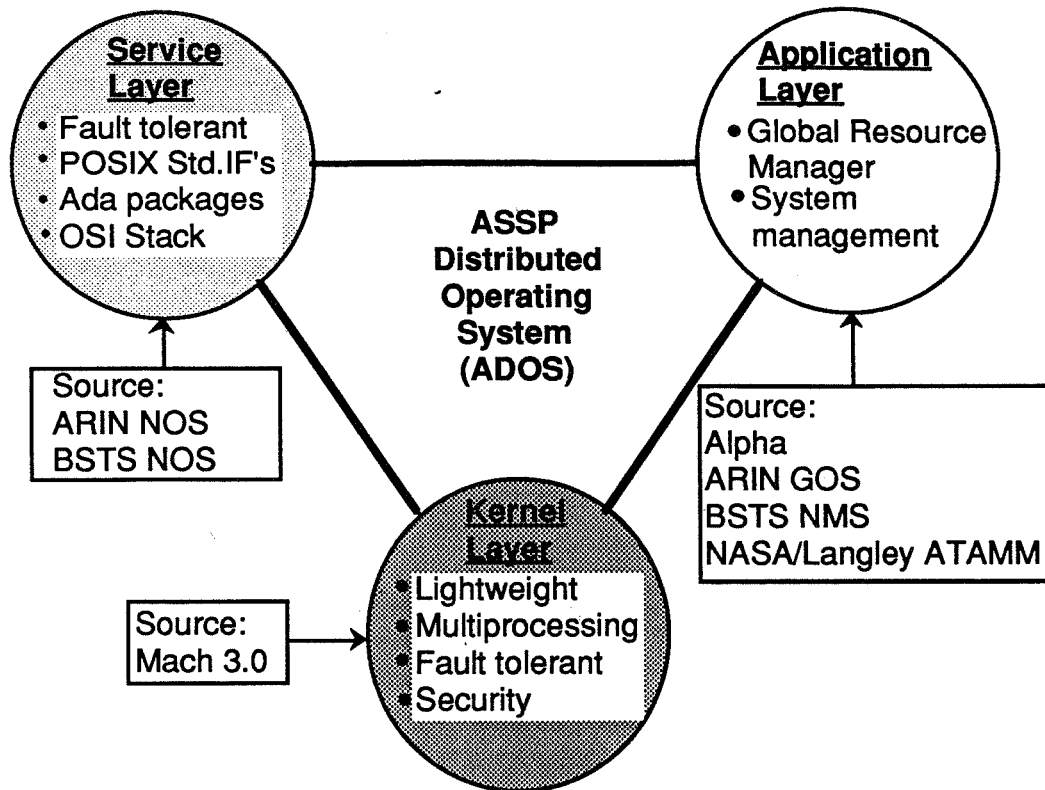


FIGURE 4-1: ASSP Operating System Diagram

The kernel layer of the ASSP O/S is very "lightweight," allocating all functions not absolutely required to control the hardware to the service layer. This contributes significantly to the portability of the kernel to other hosts. When higher performance hardware is required, the O/S can be ported to the new hardware, the Ada application software recompiled, and the upgraded system is ready for integration and test.

The ASSP program also includes fault tolerance in the O/S from the initial specifications to implementation. The system configuration design to meet the reliability and long life will be amply supported by the controlling software.

5.0 SOFTWARE DEVELOPMENT ENVIRONMENT

ASSP will develop a software development platform (SDP) that will support the development of distributed applications targeted for the ASSP environment. State-of-the-art Computer-Aided Software Engineering (CASE) tools will be provided to aid the developer in working with the ASSP O/S and embedded systems OSI profile. Central to the SDP will be the use of a common "software backplane" that provides a framework for building and managing an integrated software development environment.

Features include a common data repository, repository services, and a standard user interface. Since all data stored in the data repository can be accessed through standard interfaces provided by the software backplane, integration of commercially available tools and easy replacement/addition of new tools is supported. This is particularly important in support of upgradeability and lower life cycle cost for long-life

programs. Given a 15-20 year program lifetime, it is vital that the SDP support the insertion of new tools/technologies. By conforming to standards in this area, the ASSP SDP insures that developers are not "stranded" in the development cycle using old/obsolete tools.

6.0 ASSP SYSTEM SIMULATION SOLUTIONS

One of the principal ASSP outputs is a global, all inclusive simulation of the interaction of surveillance communications subnetworks. This simulation will be used to verify protocols, fault tolerance, network capacity, etc. Communications interactions are probably the most fundamentally difficult processing problems to analyze without simulation.

Space based as well as ground application programs can benefit greatly from the ASSP simulation tool technology development. ASSP will provide model library, user interface, and inter-simulation data transfer enhancements to the most popular commercially supported communications simulations packages. The enhancements will be fully documented and supported so that contractors will have a user friendly, highly capable simulation available which already has detailed library models of all the relevant system interconnection components and protocols. Availability of the ASSP documented and verified simulation tool allows communications simulations performed by multiple contractors and/or by multiple system elements to be combined to form large multi-system simulations.

7.0 CONCLUSION

The ASSP program directly responds to common government/industry deficiencies in providing architecture profiles that can support and upgrade processing systems without major redesigns, and procurements. It responds as well to the directives and goals for Corporate Information Management (CIM), Modular Open System Architecture Standards (MOSAS) and the Common Communication Components (Com³). This program provides capabilities to launch processing networks that are versatile, offer various levels of complexity and are capable of rapid upgrades in mission profiles, hardware, and operating systems. The capability to incorporate commercial hardware breakthroughs along with their respective software support in a very short time frame and with a minimum of redesign//retooling provides significant Life Cycle Cost (LCC) savings and is most beneficial and advantageous to the DoD.

Feasibility demonstrations with preliminary System/Segment Design Documentation deliverables are scheduled at the completion of Phase 1a (FY92). Phase 1b and Phase 2 can be regarded as options to be exercised as directed by the ASSP Program Office.



561
27236
16
1995 107747
188

A GENERIC ARCHITECTURE MODEL FOR SPACE DATA SYSTEMS

RICHARD B. WRAY

ABSTRACT

A Space Generic Open Avionics Architecture (SGOAA) was created for the NASA, to be the basis for an open, standard generic architecture for the entities in spacecraft core avionics. Its purpose is to be tailored by NASA to future space program avionics ranging from small vehicles such as Moon Ascent/Descent Vehicles to large vehicles such as Mars Transfer Vehicles or Orbiting Stations. This architecture standard consists of several parts: (1) a system architecture, (2) a generic processing hardware architecture, (3) a six class architecture interface model, (4) a system services functional subsystem architecture model, and (5) an operations control functional subsystem architecture model.

This paper describes the SGOAA model. It includes the definition of the key architecture requirements; the use of standards in designing the architecture; examples of other architecture standards; identification of the SGOAA model; the relationships between the SGOAA, POSIX and OSI models; and the generic system architecture. Then the six classes of the architecture interface model are summarized. Plans for the architecture are reviewed.

BIOGRAPHY

Richard B. Wray has a dual MS/MBA in Systems Management, Acquisition, and Contracting; a MBA with Distinction Honors in General Management; a MS in Systems Engineering and a BS in Mathematics. He is the Vice President-Technical of the National Council on Systems Engineering (NCoSE) Texas Gulf Coast Chapter, and Chairman of the NCoSE Systems Engineering Process Working Group to define a nationally recognized systems engineering process.

Mr. Wray has been with Lockheed Corporation for over eight years as an Advanced Systems Engineer responsible for developing and implementing plans, performing systems engineering and analysis, and implementing systems. He is leading the definition of a Space Generic Avionics architecture and simulation using advanced computer aided systems engineering (CASE) Tools. Prior to this, he performed and led analyses of system operational concepts, functional and performance requirements, hardware/software trade-offs and system timelines for advanced avionics systems and architectures. Mr. Wray was the lead software systems engineer for the definition and development of a software system architecture, software analyses and the DoD-STD-2167A specifications for an avionics subsystem and its interfaces to other avionics on the Advanced Tactical Fighter.



Overview SPACE GENERIC OPEN AVIONICS ARCHITECTURE (SGOAA)

for the Mission and Safety Critical Systems
Symposium

28 October 1992

Dick Wray
John Stovall

Notes:

- What we are trying to do
- What is an architecture reference model
- What is the Space Generic Open Avionics Architecture (SGOAA) Model
- Summary

Acknowledgements

The architecture presented here was developed by David Pruett of NASA's Johnson Space Center, Dick Wray and John Stovall of the Lockheed Engineering and Sciences Company (LESC). Additional support was provided by Ben Doeckel of LESL.

**GOAL: TO CREATE AN ARCHITECTURE *BLUEPRINT*
FOR TAILORING TO NEW PROGRAMS**

What Are We Trying to Do?



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

- **Objectives for developing an Architecture Reference Model:**
 - Provide an advanced architecture model as a reference for starting systems design
 - Use standards for architecture; applying interface standards is implementation specific based on mission requirements
 - Provide generic, atomic data system functions for reusing software and hardware technology in data system design
- **Results to date:**
 - *Generic System Architecture* with explicitly identified functional blocks and interfaces
 - *Generic Functional Architecture* with explicitly identified generic, atomic functions for Space Data System software
 - *Architecture Interface Model* with concept and explicitly defined interface class structure
 - Potential standards identified for use with the architecture

Doc: Wray-6, 10/20/90

Notes:

What is an Architecture Reference Model?

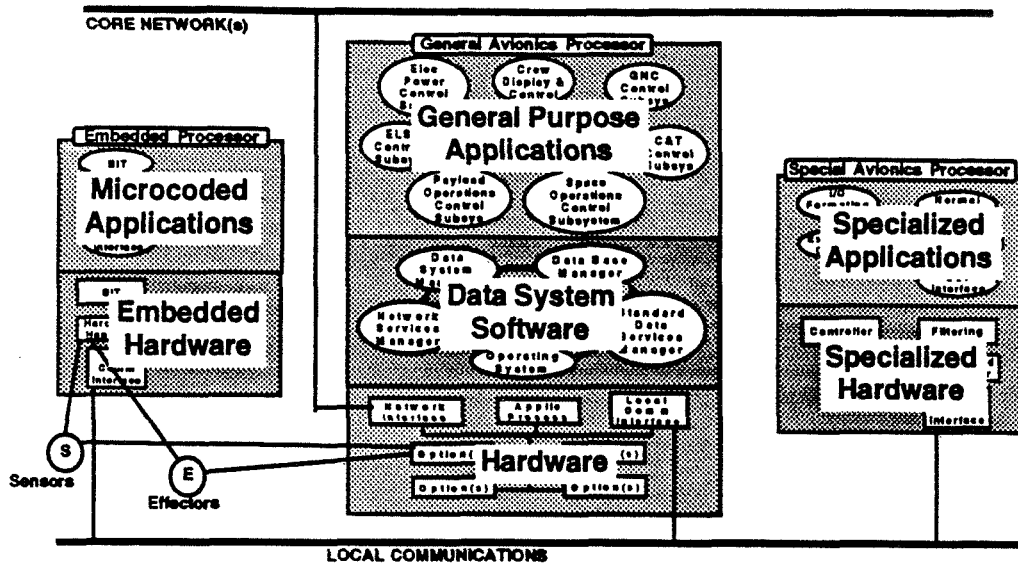
- **Architecture**
 - A set of black boxes, interfaces between the black boxes and interfaces from them to the external environment
 - All functions and performance defined at the interfaces between the black boxes
 - Software "black boxes" as well as hardware black boxes
- **Physical Interfaces**
 - Interfaces identified with physical connectivity between black boxes (hardware and software)
 - Interfaces handle signal and data flow between the black boxes
- **Logical Interfaces**
 - Interfaces identified with the meaning of data passed between two black boxes where one is the originator and one is the user of the data
 - Information exchange can cross many physical interfaces for one logical interface, and therefore cannot be identified with any one physical interface

What is the Generic System Architecture Model ?



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed



Doc Wray-3, 10/28/88

Notes:

- The generic and open system architecture proposed consists of processors which are standard, processors which can be tailored to users applications and missions needs, multiple communications mechanisms, and specialized hardware operating over standardized interfaces to the processors which manipulate the data they receive or provide.
- There are three types of processors interconnected by two types of communications. The processors provided in the architecture are the General Avionics Processor (GAP) for general purpose processing, the Special Avionics Processor (SAP) for specialized processing support, the Embedded Processor (EP) for the execution of high speed processing within the sensor and effector devices.
- The communications provided are two types: core networks for interconnecting sets of general processors or nodes, and local communications for interconnecting EPs and SAPs with their supported GAPs and general purpose processing applications.
- There are sensors and effectors which can either interact directly with the main processors (the GAPs) or indirectly through the EPs built into the sensors and effectors (if applicable).

Architecture Interface Model Classes



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

- 1 **Hardware-to-Hardware Physical**
 - Hardware Physical Interfaces - Direct Bus Connections, Memory Chip Structure
- 2 **Hardware-to-System Software Physical**
 - Hardware Registers To Software Service Drivers - Physical Interfaces
- 3 **System Software-to-Software (Local) Physical**
 - Operating System to other Local Code - Physical interfaces between Service Code
- 4 **System Software-to-System Software Logical**
 - System Services to Other Services - Logical Data Transfers from Source to User Service
- 5 **System Software-to-Applications Software Physical**
 - System Services to Local Applications - Physical Interfaces between Code Sets
- 6 **Applications Software-to-Applications Software Logical**
 - Between Software Applications - Logical Data Transfers from Source to User
 - Between Systems - Logical Interfaces for Overall Command And Control
(Mission Operational Layer)

Doc: WWS-A, 10/2002 Revised: 01/2003

Notes:

- The architecture interface model focuses on logical and physical interfaces to isolate and enable specifications of effective interfaces. Logical interfaces are those connecting elements which provide information with those needing it; these are "ought to" type interfaces not real interfaces. So hardware logical does not exist because either it connects or it does not, no "ought to" is involved. This progression from hardware to system logical interfaces moves from the things people can touch and feel to the conceptual/logical. Software physical interfaces are those interfaces between two sets of software code, e.g., one code package calling another.
- Class 1, Hardware-to-Hardware Interfaces (Physical), addresses hardware component modularity and portability, and maintainability and technology upgradability of platform hardware over extended space avionics life cycles. These hardware interface standards must be definitive as to the software driver interface requirements needed to communicate with that hardware.
- Class 2, Hardware-to-System Software Interface (Physical), is the Operating System (OS) hardware driver software to invoke platform services internal to the Application Platform. Each of these standards must specify the software interface binding requirements for "plugging" a driver into the OS.
- Class 3, System Software-to-Software (Local Physical), is provides access from the operating system to all other platform services and applications in support of application portability.
- Class 4, System Software-to-System Software Interfaces (Logical), is the internal interface for transfer of data between Application Platform Data System Services. For example, this is the logical interface between the Data Base Manager in an Application Platform communicating with the Data Base Manager in another platform.
- Class 5, System Software-to-Application Software Interfaces (Physical), is defined primarily to support Application Software portability.
- Class 6, Application Software-to-Application Software (Logical), consists of application-to-application software interfaces for both local/node and other systems. Applications software interfaces are the internal interfaces for transfer of data between Application Software within an Application Platform. These are also the external logical interfaces between Application Software on the Application Platform with Application Software on other Application Platforms.

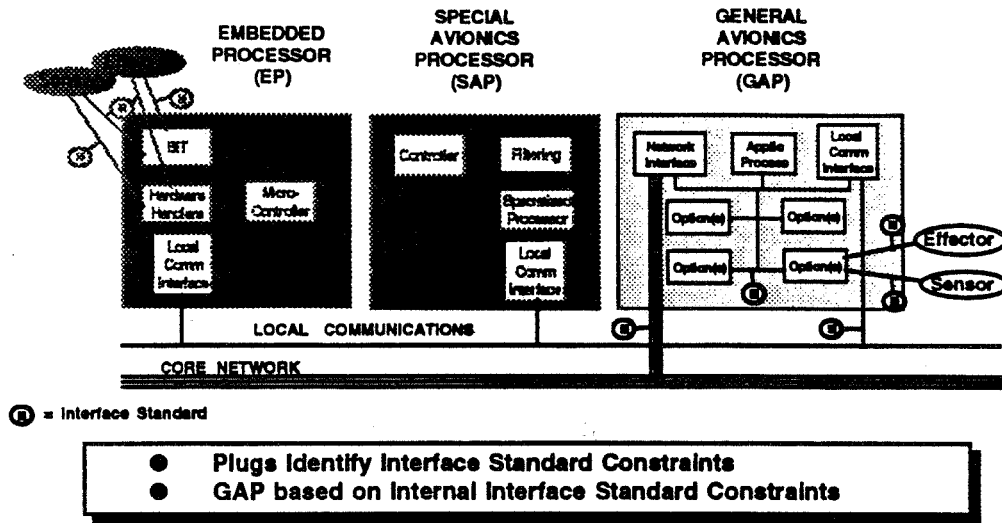
Hardware-to-Hardware Physical Interfaces



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

Class 1. Hardware Application Platform

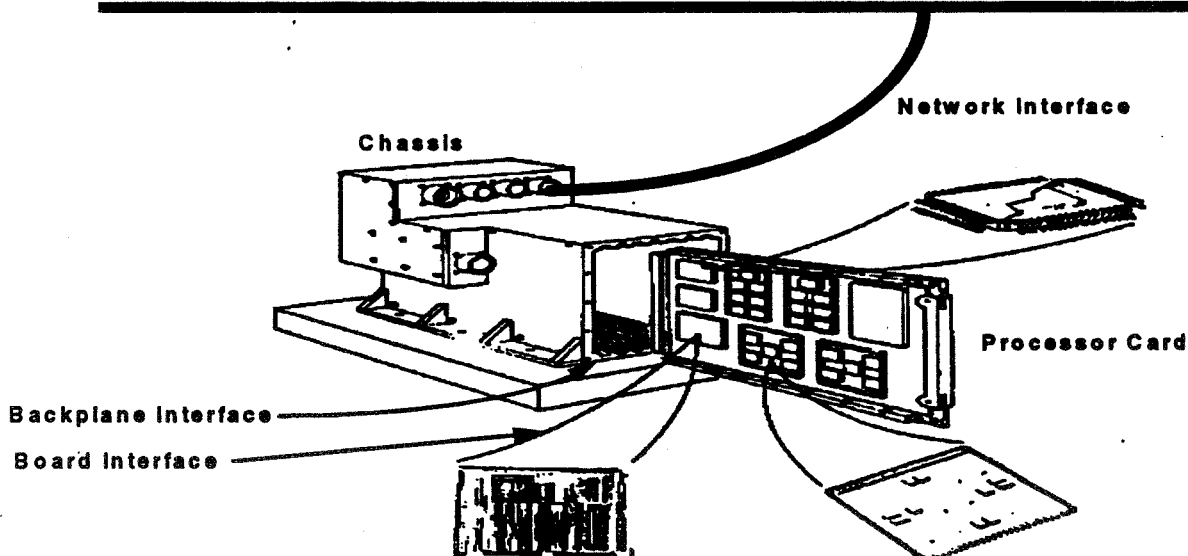


Doc: Wway-4, 14/02/92

Notes:

- The hardware to hardware physical interfaces are shown in this slide. These interfaces consists of the nuts, and bolts, chips and wires of the hardware architecture described previously. With regard to the model, it consists of all the hardware to hardware interfaces within each processing element, as well as the hardware interfaces to the external environment by way of the core network, local communications or direct interfaces. The focus in this standard is on GAPs which provide the greatest flexibility in configuring the system to accomplish different purposes in avionics. The GAP includes hardware components to interface to a core network, to interface to local buses, to process applications, and optional components for other purposes (such as serial input and output to direct analog and discrete links). As implied by the darker shading on the GAP, the GAP is the focus of efforts to standardize the hardware processor support due to its general purpose nature. An example of such hardware interfaces is shown below.

Core Network



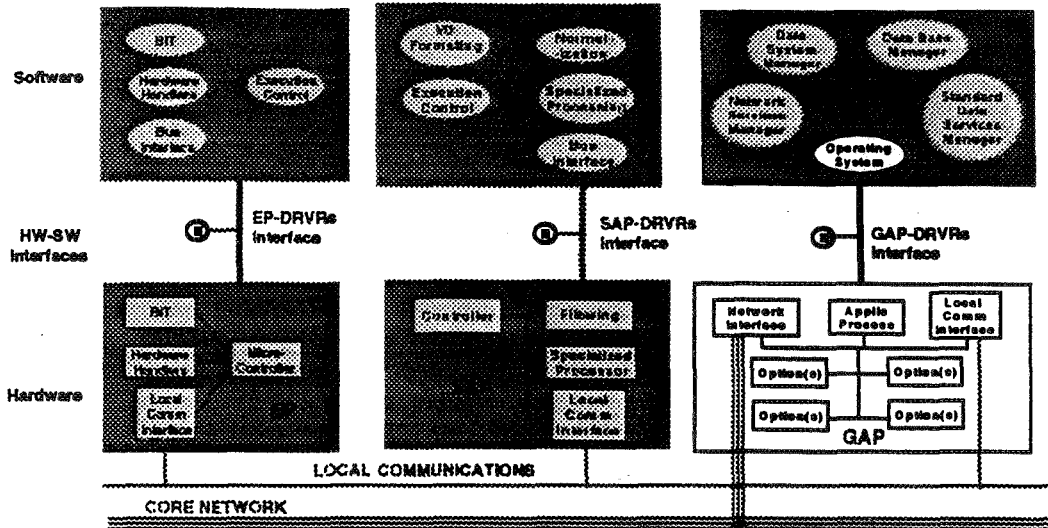
Hardware-to-System Software Physical Interfaces



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

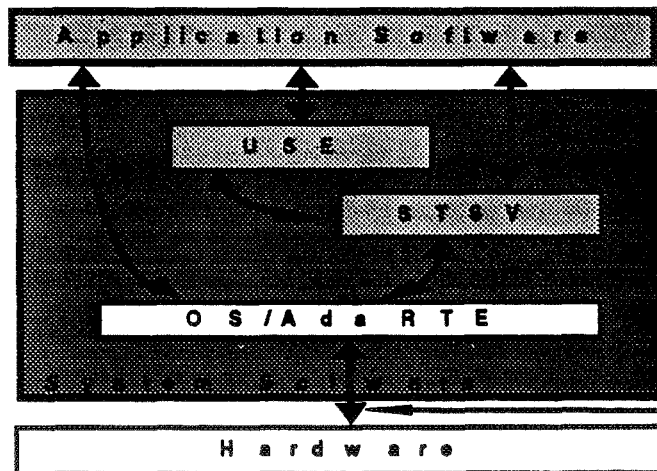
Class 2: Application Platform Hardware to Service Drivers



Doc Wray-A, 10/26/88 Revise: 01/89

Notes:

- The hardware to system software interfaces are shown in this slide. These consist of the interfaces from the system software drivers (i.e. in the OS, data system manager, etc.) to the hardware instruction set architecture (ISA) and register usage. With regard to the model it is internal to each processing element. The grayed out elements are a repeat of the previous figure; the white elements represent the new capabilities and interfaces added by this class. This class provides low level software drivers to interact with the hardware for each of the processor types (EPs, SAPs, and GAPs). The drivers are (obviously) hardware dependent, but this enables the architecture to begin to partition out the hardware dependencies, which is a key in providing for technology upgradability in the future. All the drivers for all processor types are contained in the Space Data System Services (SDSS) sub-architecture.
- The system services software for the GAP are organized into five categories. These categories are the Data System Manager, Data Base Manager, Standard Data Services Manager, Operating System, and Network Services Manager. Note the naming convention used for the GAP hardware to the OS drivers, i.e., GAP-DRVR. This single link will be broken open in the next figure to show its component elements. An example (drawn from the Space Station Freedom) of software driver interfaces is shown below.



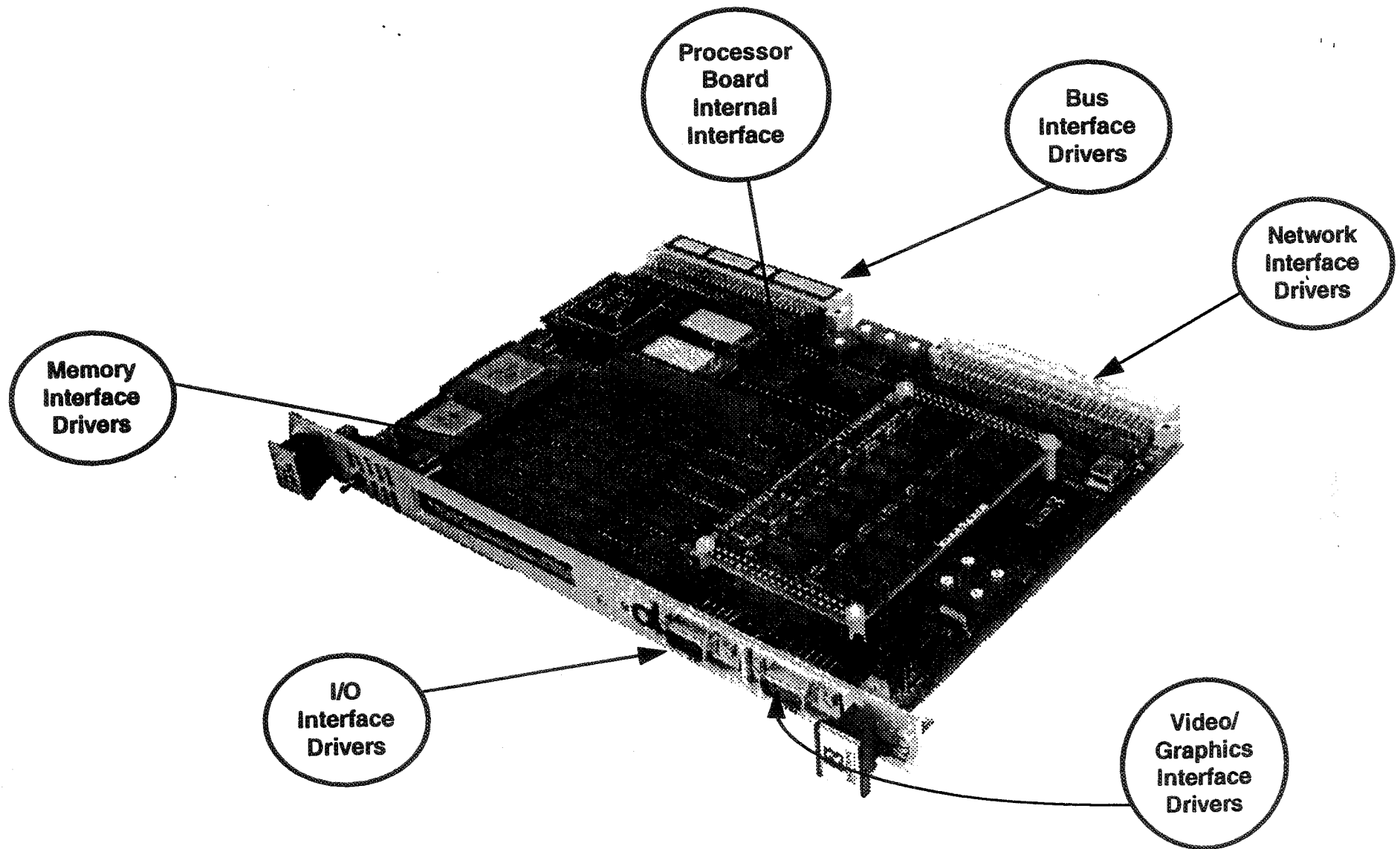
Class 2

Example of Hardware to Software Physical Interface



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed



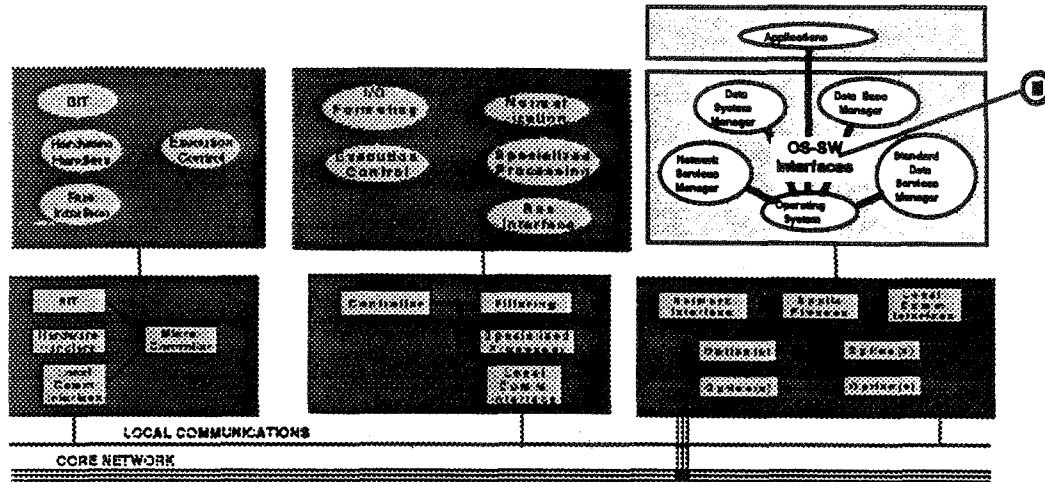
System Software-to-Software (Local) Physical Interfaces



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

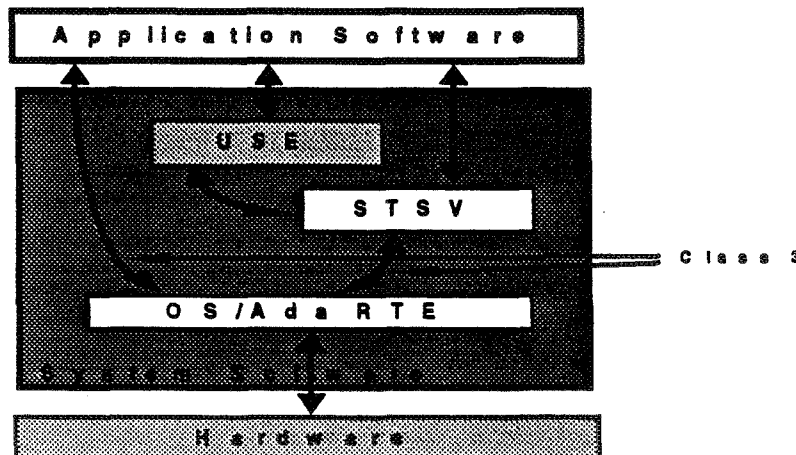
Class 3. Operating System to Other Code (i.e., Services or Applications)



Doc: Wtop-7, 10/20/82 Revised: 01/83

Notes:

- The system software drivers to local system software service interfaces are shown in next. These consist of the Input/Output handler calling conventions and context switch conversions between the system software drivers on one processing element interfacing with one or more system software services to provide for local information exchange. The grayed out parts of the figure represent the material covered in Classes 1 and 2, the white parts of the figure are the new material added in Class 3. Since Class 2 provided the software drivers to isolate the hardware, Class 3 provides the remainder of the local software services needed to operate the computer system. They all fall into the Space Data System Services (SDSS) sub-architecture, consisting of the Data System Manager, Data Base Manager, Standard Data Services Manager, Operating System, and Network Services Manager. Class 3 provides all remaining services and the interfaces between the local services for effective local interprocess communications and support. These interfaces are physical interfaces because they enable software service code to interact with software service code in other local entities. Class 3 interfaces meet derived requirements based on the need of an application to support users.
- The naming convention (e.g., OS-SW) is shown in this figure to indicate all the OS links both down to OS drivers and up to other high level processes will be identified explicitly by their names. An example from the Space Station Freedom project is shown below of these interfaces.



7
A.

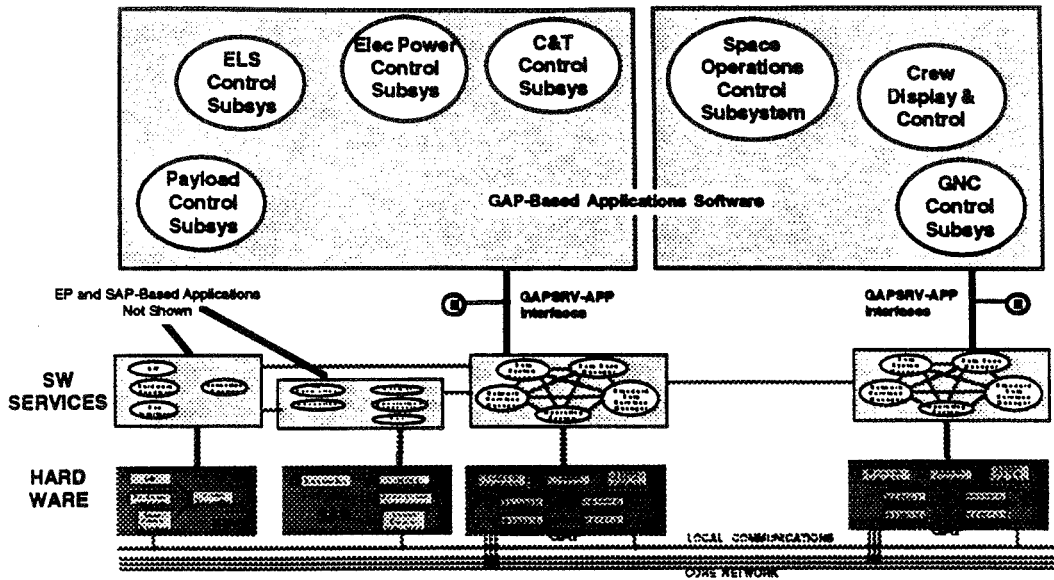
System Software-to-Applications Software Physical Interfaces



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

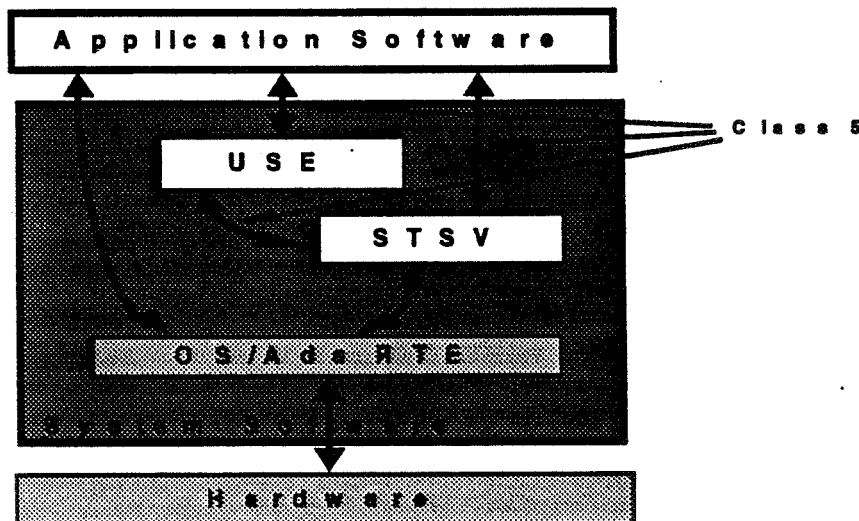
Class 5. Services to Applications



Doc: Wwp-4, 10/24/82 Revised: 5/24/82

Notes:

- The system software services to applications software interfaces are shown here. This is the physical interface within a processing element between the application software and the system software (language bindings/specification) to allow provision of needed services. The grayed out parts of the figure represent the material covered in Classes 1 to 4, the white parts of the figure are the new material added in Class 5. Since Classes 1 to 4 isolated the hardware and software services in all the processors, Class 5 adds the interface capability for services in any processor to interact with an application executing in the processor; this provides the basic multi-processor capability to meet actual user requirements in processing. Applications can operate in any GAP, with potential partitioning of an application across multiple GAPs. Similarly, applications can operate in any SAP or any EP. These interfaces are physical interfaces because the applications software code is interacting with the service software code. Class 5 interfaces meet derived requirements based on the need of an application to support users in a multi-processing environment.
- The naming conventions identify the higher level interfaces which will be specified in more detail in lower level diagrams. An example of these interfaces from Space Station Freedom is shown below.



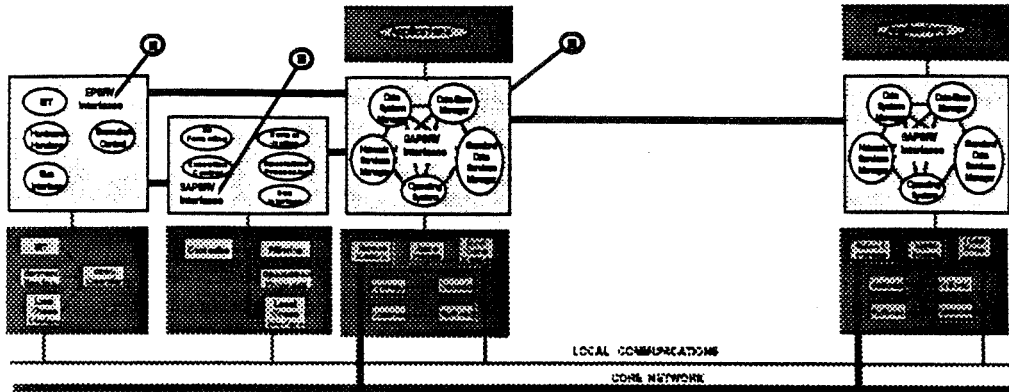
System Software-to-System Software Logical Interfaces



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

Class 4. Services to Other Services Data Transfers

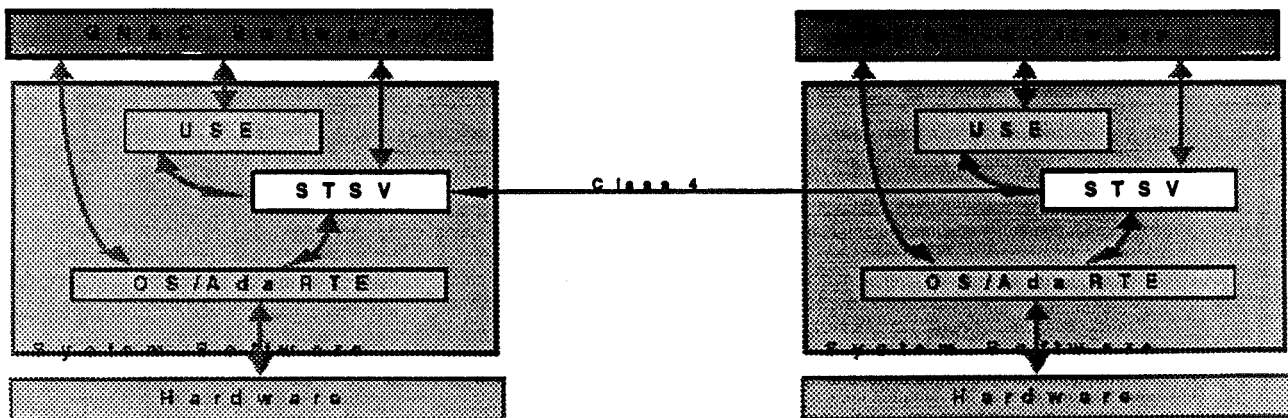


- Lateral Interfaces between Service Using Data and Service Generating Data
- No Logical Architectural Difference between Local and Remote Services Access
- Meets Requirements for Applications Support - A Derived Requirement

Doc: Wp-8, 102882 Revised: 8/1988

Notes:

- The system software services to remote system software interfaces are shown in this slide. This is the peer to peer interface of system software in one processing element (GAP, SAP or EP) interfacing with the system software in an external processing element to coordinate operations in a distributed environment. The grayed out parts of the figure represent the material covered in Classes 1 to 3, the white parts of the figure are the new material added in Class 4. Since Classes 1 to 3 isolated the hardware and software services in each processor, Class 4 adds the interface capability for services in one processor to interact with services in another processor; this is the heart of multi-processor capability needed in modern space avionics systems. GAP services can interact with EP and SAP services and other GAP services. These interfaces are logical interfaces because the service originating data is interacting with the service that will use the data (i.e., that will transform the data into another form for a purpose). Class 4 interfaces meet derived requirements based on the need of an application to support users in a multi-processing environment.
- The GAP to services interfaces are defined by the naming convention as GAPSRV- to indicate that GAP services would be specified by the standard interface specification, and could be broken out by subsequent lower level charts. An example from the station is shown below.



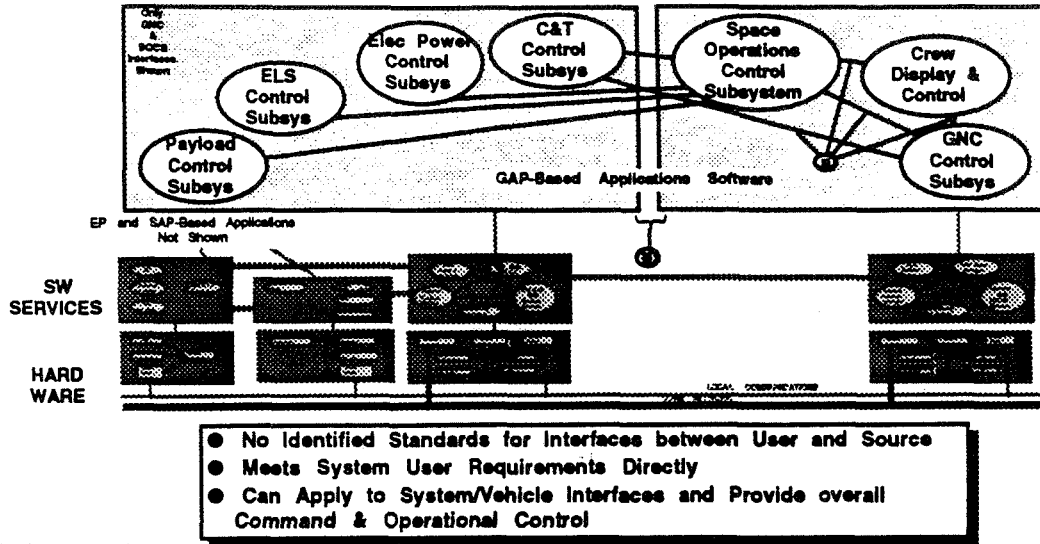
Applications Software-to-Applications Software Logical Interfaces



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

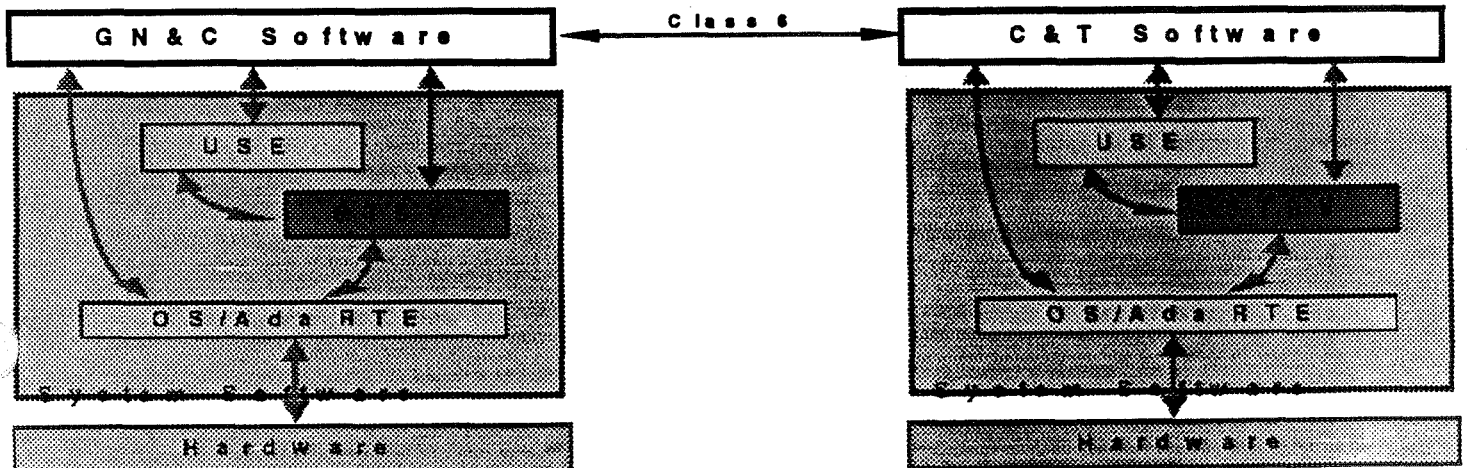
Class 6. Applications to Applications Logical Data Transfers



Doc: Wwp-1A, 140892 Rev. 02/02

Notes:

- The applications software to applications software interfaces are shown. This peer to peer information exchange and coordination interface between application software modules. Applications do not communicate directly; hence this is a logical interface. All communication is through a Class 5 (P) standard interface to System Services which provides the physical communications path between applications. This interface may be between applications within a processing element or between applications in separate processing elements. The grayed out parts of the figure represent the material covered in Classes 1 to 5, the white parts of the figure are the new material added in Class 6. Since Classes 1 to 5 isolated the hardware, software services and applications in any processor, Class 6 adds the interface capability for an application in any processor to interact with another application executing in any processor; this provides the basic multi-processor capability to meet multiple actual user requirements in processing. Applications can operate in any processor (i.e., GAP, SAP or EP), with cooperating interactions to support the needs of the users. The interfaces are logical interfaces because the application originating data is interacting with applications that will use the data (i.e., that will transform the data into a form useful to the user or to another application for a user's ultimate purpose). Class 6 interfaces meet user and derived requirements based on the need of multiple applications to support users in a multi-processing environment. The example below is provided from the station.





FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

Current Status and Plans

- **Review by SATWG and others:**
 - Review by Space Avionics Architecture Panel
 - Planned for presentation to other forums
 - Published in AFCEA's Signal Magazine (September)
 - ⇒ Mission and Safety Critical System Symposium (October 28)
 - SIMTEC invitation (November 4)
 - SATWG and SAAP Software Workshop (November 18)
 - SAE invitation (November)
- **Enhancements in the works:**
 - FDIR/RM requirements to be incorporated
- **Applied to projects:**
 - Used in Artemis Common Lunar Lander space data system
 - Beginning to build StateMate dynamic model (simulation)
 - Beginning point for institutional analysis and design of flight data systems

Doc: Wwp-11, 10/20/98

Notes:

- The Reference Architecture Model Must Be Based on Standards
- It Must Span Platforms for All Missions and Operational Requirements
- A Space Generic Open Avionics Architecture Must be Adaptable
- Avionics Control Structure Must be Integrated in an Architecture
- Support Tailoring to Multiple Missions
- The advanced avionics architectures must fit and extend the POSIX Open Systems Environment model
- The space generic avionics functional architecture was successfully applied to the Common Lunar Lander, with a preliminary design for the data system in 2 days
- The architecture interface model makes an explicit and rational model of how hardware and software interfaces should be defined
- A common advanced architecture for all future space platforms is feasible and achievable



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

Addenda

Doc. Wray-13, 102592

Notes:

Examples of Architecture Reference Models



FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

- International Standards Organization (ISO) Open Systems Interconnect (OSI) 7 Layer Reference Model
- National Institute of Standards and Technology (NIST) Portable Operating System (POSIX) Open Systems Environment (OSE) Reference Model
- *Proposed* Space Generic Open Avionics Architecture (SGOAA)

The objective of a reference model is to identify
INTERFACES
between
FUNCTIONAL BLOCKS
so that existing and future
STANDARDS
can be applied at the
INTERFACES
in a systematic way to meet mission requirements

Doc: W99-14, 10/20/99

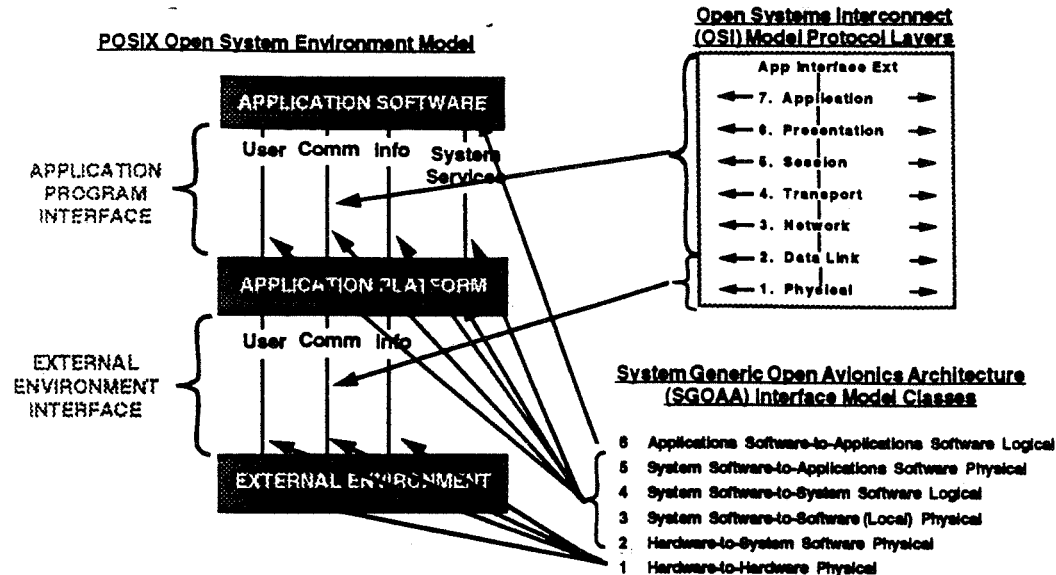
Notes:

Relationships between OSI and SGOAA Models



FLIGHT DATA SYSTEMS DEPARTMENT

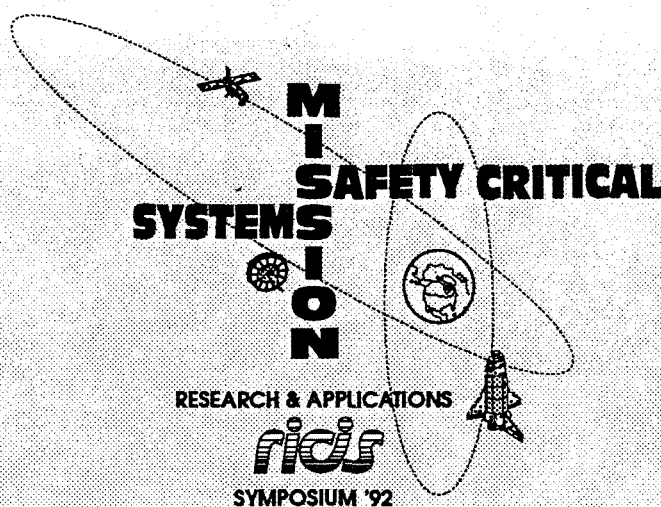
Lockheed



Dick Wray-15, 10/26/92 Revised: 6/1/98

Notes:

- The POSIX Open System Environment (OSE) Reference Model is the basis for the generic and open avionics architecture models, and for application software portability and interoperability. It can be related to the Open System Interconnect (OSI) model and the SGOAA interface model as shown in this slide. The OSE model communications link protocols are defined in detail by the OSI model for peer-to-peer communications. The OSE model interface classes are defined in detail by the SGOAA.
- The OSE Reference Model enables application software portability at the source code level and application software and system service interoperability between heterogeneous systems. Definition of entities and interfaces based on the OSE model can facilitate requirements definition for designs which have the open and generic characteristics needed.
- There are three types of entities used in the OSE Model: Application Software, Application Platform and External Environment. There are two types of interfaces: the Applications Program Interface and the External Environment Interface.
- The OSI Reference Model is a Network Services Model. Network Service is only one resource of many competing resource processes provided by both the POSIX and SGOAA Models. Applications gain access to POSIX Network Services via the POSIX API Communications Services Interface and to SGOAA Network Services via the SGOAA Class 5 Interface (Applications Software-to-System Services Software). In the OSI model, applications gain access to Network Services via an applications-to-services interface. Interfaces provided by Network Services must be open network interfaces, protocol independent and provide for network protocol interoperability. The POSIX OSE reference model focuses on the requirements of application portability and system interoperability at the source code level by addressing these objectives at the Applications Program Interface (API) and at the External Environment Interface (EEI). Internal Application Platform Interfaces are not addressed.
- The OSI model may be mapped into just the communications links of the POSIX OSE model API and the EEI to define the communications protocols. The SGOAA model may be mapped into the user, communications, information, and systems services links of the POSIX OSE model to define the content of all the interfaces. Thus, the three models are complementary.

27236
P. 20

1995107748

504057

22P.

EVOLUTIONARY TELEMETRY & COMMAND PROCESSOR (TCP) ARCHITECTURE

Mr. John R. Schneider

ABSTRACT

Current development is underway to build a low cost, modular, high performance, and compact Telemetry And Command Processor (TCP) as the foundation of command and data handling subsystems for the next generation satellites. The TCP product line will support command and telemetry requirements for small to large size spacecraft and from low to high rate data. It is compatible with the latest TDRSS, STDN, and SGLS transponders and provides CCSDS protocol communications in addition to standard TDM formats. Its high performance computer provides computing resources for hosted flight software. Layered and modular software provides common services using standardized interfaces to applications thereby enhancing software re-use, transportability, and interoperability. The TCP architecture is based on existing standards, distributed networking, distributed and open system computing, and packet technology. The first TCP application is planned for the 94 SDIO SPAS III mission. The architecture enhances rapid tailoring of functionality thereby reducing costs and schedules during development of individual spacecraft missions.

BIOGRAPHY

Mr. John R. Schneider joined Fairchild Space in December 1991 as a Staff Engineer with the Communications, Data Handling, and Power Systems Department. He currently is working as a system engineer on the TCP hardware and software architecture. Prior to Fairchild Space, Mr. Schneider concentrated as a system engineer on satellite ground systems while employed with NASA/GSFC, NOAA, Mitre, SPACECOM, and Ford Aerospace. His ground systems experience includes RF front end stations, control centers, data processing centers, and communication networks. Notable past projects include Space Station Freedom, EOS-DIS, ERTS/Landsat, TIROS-N, and TDRSS. He holds a BSEE earned in 1968 from the University of Cincinnati.

C-2

N



**MISSION AND SAFETY CRITICAL SYSTEMS
RESEARCH & APPLICATIONS**

EVOLUTIONARY TELEMETRY & COMMAND PROCESSOR (TCP) ARCHITECTURE

**RICIS Symposium '92
University Of Houston - Clear Lake
Houston, Texas
October 28-30, 1992**

***John R. Schneider
Communications, Data Handling, & Power Systems
Fairchild Space
Department A-33
Mailstop A-14
20301 Century Boulevard
Germantown, Maryland 20874
(301) 428-6227***



THE TELEMETRY & COMMAND PROCESSOR (TCP) IS A VITAL MISSION & SAFETY SYSTEM

The RICIS Symposium '92 focuses on Mission Safety Critical Systems. These systems are characterized as having high criticality and whose correct execution is vital to the successful operation of the mission. In this symposium, these systems include computer controlled real-time applications. This Session II, Generic Architectures For Future Flight Systems, focuses upon architectures of both spacecraft and avionic control systems. This presentation describes a new product, the Telemetry And Command Processor (TCP), currently under development by Fairchild Space. The TCP will serve as the foundation of a control system for future spacecraft.

Mission success is highly dependent upon the real-time control system to reliably perform housekeeping functions, data handling, and information exchange with mission personnel. The need for higher data rates, more on-board processing power, larger storage capacities, and improved communication protocols require the development of new architectures. In addition, industry pressure to rapidly produce new spacecraft with a competitive cost require that modularity and optimum re-use concepts be used in the architecture. In recognition of these needs, Fairchild Space has started development efforts for future real-time control systems - the TCP. The TCP is based on Fairchild's heritage with spacecraft flight data systems especially in providing standardized control systems for multimission spacecraft. Also, the TCP will benefit from hardware and software Independent Research And Development (IR&D) programs. The best features of past systems are engineered into the TCP along with using state-of-the-art technologies and design concepts.

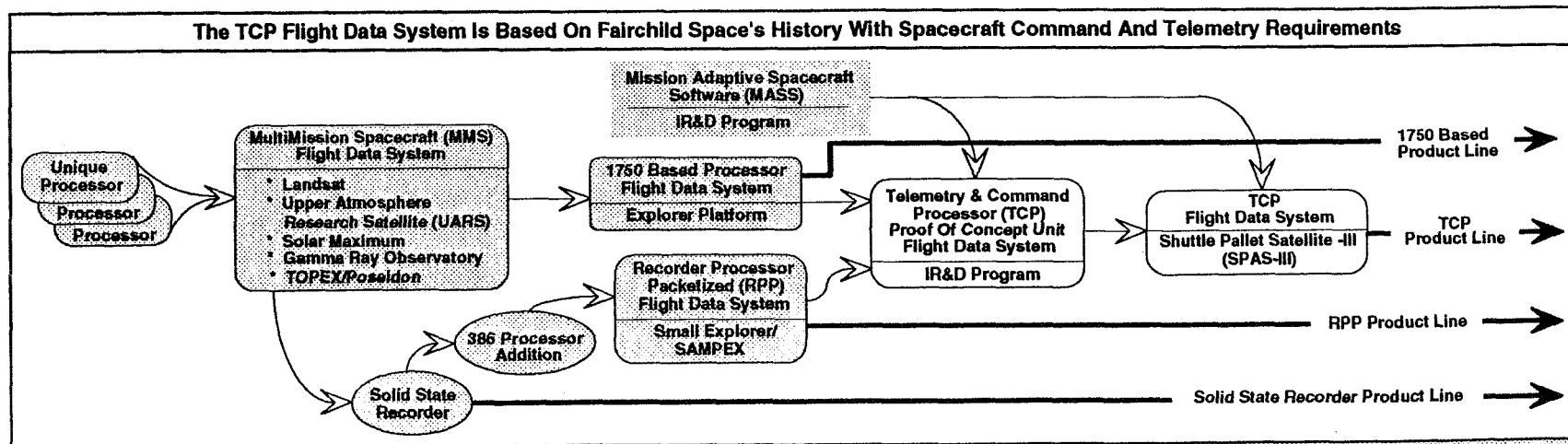
The TCP provides real-time computer based spacecraft control, data handling, and communications with other spacecraft subsystems and with mission personnel. It is compatible with many space-to-ground communication links. The communication link uses the CCSDS protocols in addition to currently used formats (e.g., TDM telemetry and NASA 48 bit command formats). Hardware/software re-use, transportability, and rapid configuration for mission-to-mission adaptability are key design drivers to the TCP. The TCP architecture uses modularity, standardized interfaces, and "information hiding" to satisfy these drivers. The TCP is configured from a toolkit of modular cards using layered software. The cards and software support open system, networking, distributed processing, and packet concepts. Mission unique functions and/or technology insertion is easily achieved through the addition of a new card(s). Reliability from mission-to-mission is also increased through the re-use of proven cards and software. In addition, re-use provides the TCP with the capability to adapt to specific missions at reduced cost and development schedule.

OBJECTIVE

Successful Operation Of The Command And Data Handling (C&DH) Subsystems Is Vital To Spacecraft Mission Execution. Next Generation Spacecraft Range From Small-To-Large Size And From Low-To-High Data Rates. They Will Use State-Of-The-Art Concepts Including CCSDS Communications, On-Board Networking, Distributed Processing, And Open System Architectures. In Addition, Budget Constraints Have Mandated Reduced Costs And Shorter Development Schedules.

AN APPROACH

The Telemetry & Command Processor (TCP) Is The Foundation Of The C&DH Subsystem. It Provides Real-Time Control/Monitor Of Spacecraft Subsystems And The Data Exchange With Mission Personnel. The TCP Supports TDRSS, STDN, And SGLS Transponders. It Provides CCSDS Protocol Communications In Addition To TDM Formats. The TCP Architecture Uses Modular Cards With Layered Software. The Architecture Supports Open Systems, Networking, Packets, And Distributed Computing Concepts. This Approach Enhances Rapid Tailoring Of Functionality And Optimum Re-Use Resulting In Reduced Costs And Schedules For Spacecraft Missions.





FAIRCHILD
S P A C E

THE TCP PROVIDES REAL-TIME CONTROL AND DATA HANDLING FOR ALL SPACECRAFT SUBSYSTEMS

Spacecraft C&DH Subsystems generally contain of one or more central processing units surrounded by peripherals. Peripherals include interface units and storage devices. C&DH Subsystems serve the overall purposes of spacecraft subsystem management, data collection, spacecraft health maintenance, and information exchange between the spacecraft and mission personnel. The central processing unit provides the computational power to perform housekeeping functions, data handling, and data communications. Interface units provide the signal conditioning, handshaking, and data transfer with the subsystems. Data storage devices are primarily used for recording on-board data for later playback due to various space-to-ground communication link outages.

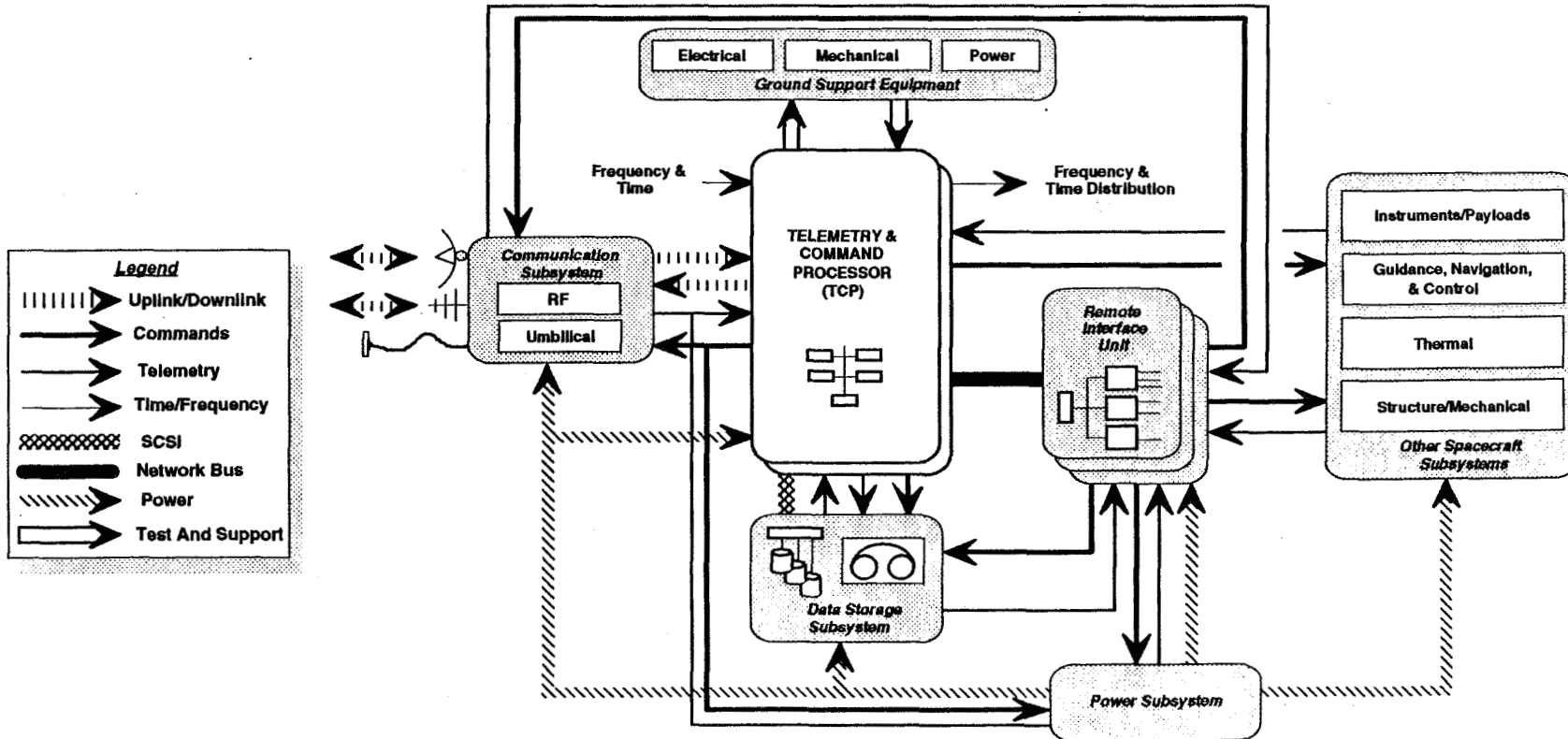
In this context, the TCP contains the central processing unit, direct subsystem interfaces, remote interface unit connection via networks, and disk/tape recorder storage device interfaces. Interfaces also exist that allow multiple TCPs to communicate among themselves where more than one TCP are used for reliability purposes and/or for distributed processing. These interfaces provide health and well-being information to the TCPs. The information flow can be across dedicated interfaces or across a networked configuration. For multiple TCPs, the TCP also contains "cross-strapping" of critical input/output interfaces so that only one may serve as a master at a time. In addition, the TCP contains interfaces for use by ground support equipment. These interfaces are used during development for "box" level testing and for the loading/verifying of flight software. The attached viewgraph provides a context view of the TCP's relationship with the spacecraft subsystems.

The TCP supports the overall purposes by reliably providing for command reception, validation, and distribution to the subsystems; the collection, formatting, and distribution of data; the storage and later retrieval of data; the maintenance of on-board time; and the general purpose computational environment to operate flight application software including attitude control, power management, and thermal management.

The TCP's Overall Purposes Are To 1) Control/Monitor Spacecraft Subsystems, 2) Collect Data, 3) Maintain Spacecraft Health, And 4) Exchange Data And Commands With Mission Personnel.

In Support Of These Purposes, The TCP Provides The Following High Level Requirements:

- * Command Reception, Validation, And Distribution
- * Data Collection, Formatting, And Distribution
- * Data Storage And Retrieval Management
- * Spacecraft Time Maintenance
- * On-Board General Processing To Host Flight Software



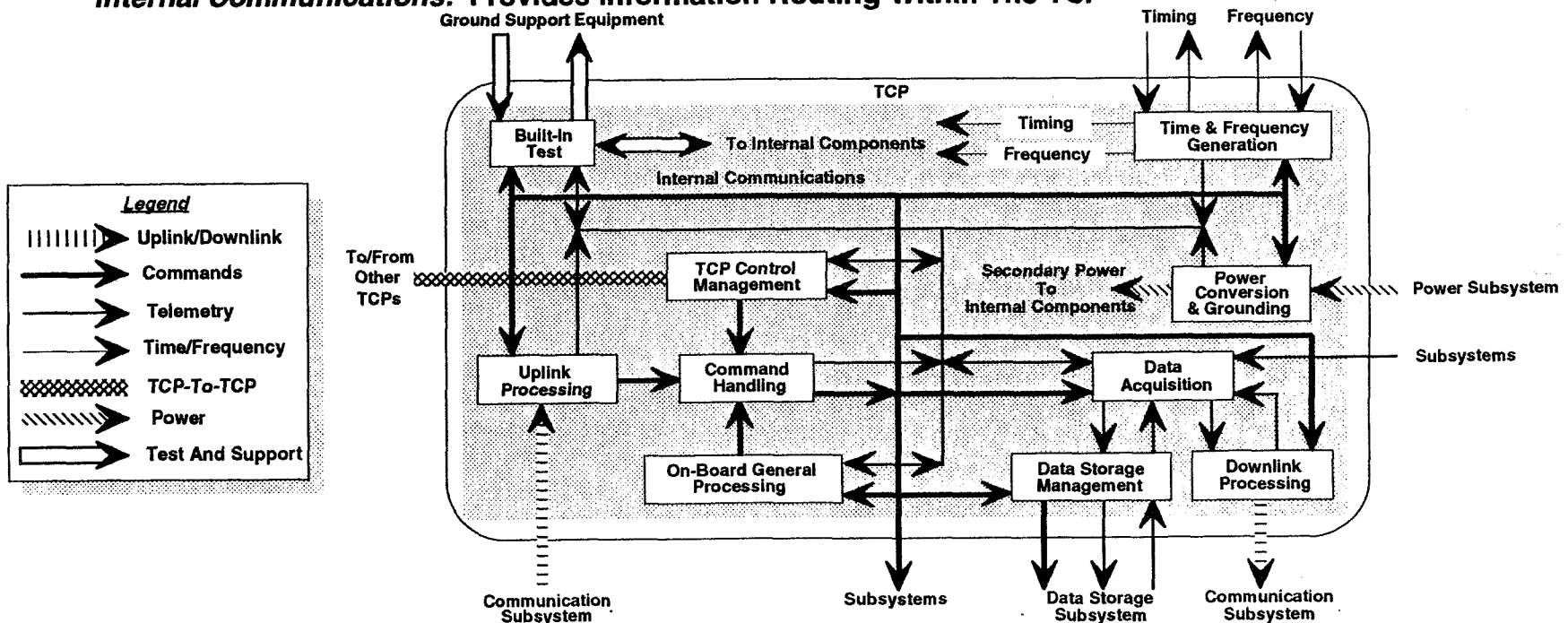
The architectural development of the TCP requires understanding of the major functions and their inter-relationships to satisfy requirements.

The TCP performs 11 major functions. **Uplink Processing** interfaces with the communication subsystem for command reception. It provides communication handshaking and synchronization, protocol processing, and command validation. **Command Handling** provides storage of procedures and time tagged commands and distributes all commands to the subsystems. **Downlink Processing** provides the communication subsystem interfaces for the transmission of data. It performs handshaking with the communication subsystem, modulation processing where appropriate, and protocol processing. **Data Acquisition** performs data collection from the spacecraft subsystems and from TCP internal functions. It also performs the routing of data to the proper destination. **Data Storage Management** performs the data transfer to/from the storage devices and storage control. For disk storage devices, it provides file manipulations (e.g., open, close, delete, copy, and move) and file management (e.g., file directory maintenance, naming, and dating). **On-Board General Processing** provides a general purpose computing environment. It hosts the various mission dependent flight software. Examples of resident software include attitude control algorithms and power resource management. **TCP Control Management** performs TCP configuration management and control of operational modes and capabilities. It also provides the TCP health and well-being information to other TCPs in a multiple TCP configuration. **Time And Frequency Generation** provides spacecraft time management and clock/frequency generation. It also performs the synchronization of time/frequency with external sources and the distribution of time/frequency to the subsystems. **Built-In Test** evaluates internal circuits for proper operation and performance. It interfaces with the ground support equipment for box/card level testing during development and pre-launch activities. It also "loads" the flight software into the TCP. **Power Conversion And Grounding** receives spacecraft primary power and generates the secondary power for the internal functions. **Internal Communications** provides the routing of all signals, power, and grounds among the functions/cards within the TCP.

An overview of the inter-relationships of these functions is illustrated in the attached viewgraph. The viewgraph illustrates two key data paths: commands and telemetry. For commands, the communication subsystem provides the uplink signal to the **Uplink Processing** function which retrieves the command information for transfer to **Command Handling**. **Command Handling** also receives command information from **TCP Control Management** and **On-Board General Processing**. **Command Handling** processes the commands for distribution to the subsystems. It also stores, where appropriate, procedure and time tagged commands. For telemetry, **Data Acquisition** collects data from the subsystems and internal functions. It routes the data to four destinations: 1) **TCP Control Management** receives internal telemetry for monitoring TCP operations, 2) **On-Board General Processing** receives data for input to the resident flight application software, 3) **Data Storage Management** receives data that is to be stored, and 4) **Downlink Processing** receives data to be formatted for downlink transmission.

The TCP Performs 11 Major Functions Supporting The High Level Requirements. These Functions Are:

- * **Uplink Processing:** Provides Command Reception And Validation
- * **Command Handling:** Provides Command Decoding, Storage, And Distribution
- * **Downlink Processing:** Provides Data Formatting And Distribution To Communication Subsystem
- * **Data Acquisition:** Provides Data Collection From Subsystems And Data Routing
- * **Data Storage Management:** Provides Tape Recorder/Disk Interfaces For Data Recording And Playback
- * **On-Board General Processing:** Provides General Computational Resource For Flight Applications Software (e.g., Attitude Control And Power Management)
- * **TCP Control Management:** Provides TCP Configuration And Operational Control
- * **Time And Frequency Generation:** Provides Spacecraft Time Maintenance
- * **Built-In Test:** Provides Self Checking Tests And Diagnostics And Ground Support Equipment Interfaces
- * **Power Conversion And Grounding:** Provides Primary Power Conversion
- * **Internal Communications:** Provides Information Routing Within The TCP

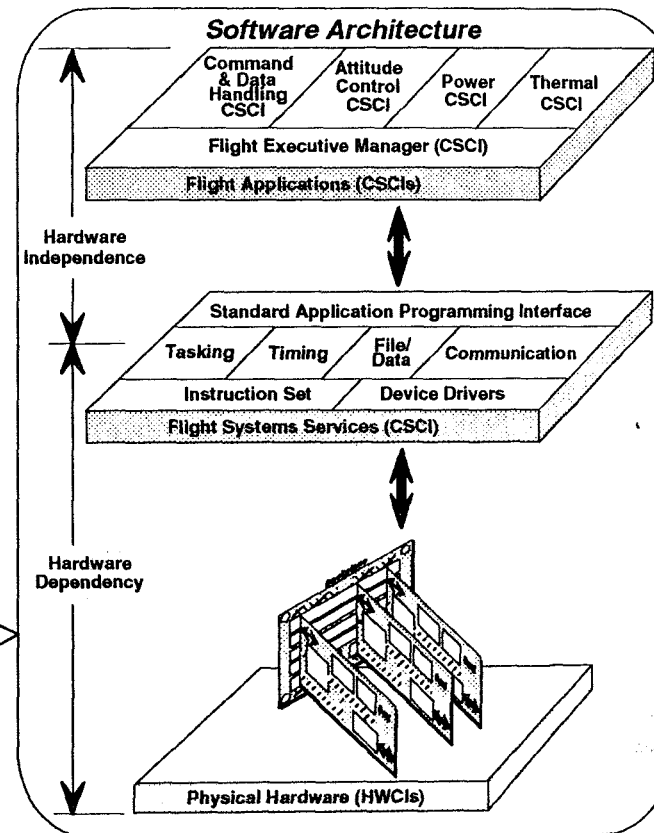
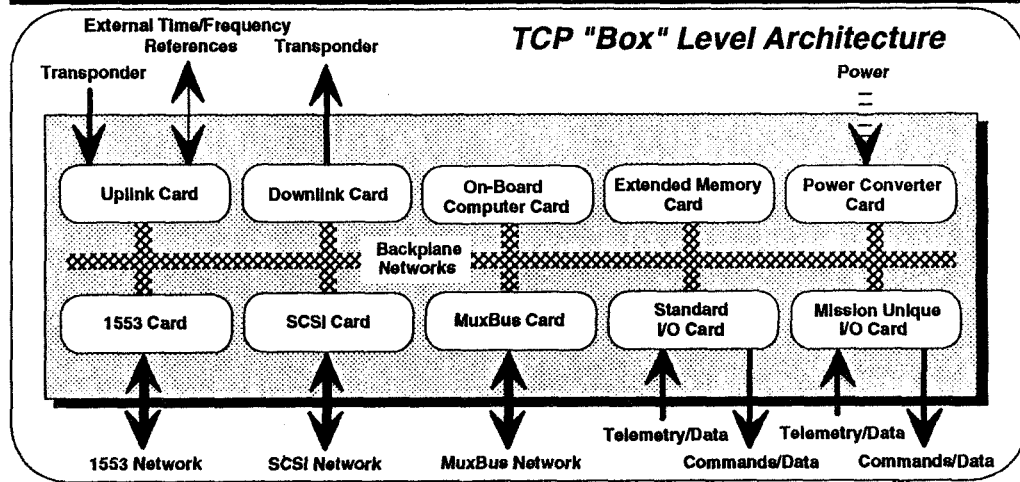


Key design drivers considered during the development of the TCP architecture include hardware/software re-use, transportability, rapid adaptability from mission to mission, modularity, and standardized interfaces. Analysis of the eleven major functions and the design drivers has resulted in an architecture consisting of modular cards connected together via a backplane network within the TCP enclosure. The currently defined card set is illustrated in the attached viewgraph and consists of 10 cards: 1) Uplink, 2) Downlink, 3) On-Board Computer, 4) Extended Memory, 5) Power Converter, 6) 1553 Network I/O, 7) SCSI, 8) MuxBus Network, 9) Standard I/O for direct command and telemetry, and 10) Mission Unique I/O.

The backplane architecture, illustrated in the viewgraph, provides the wiring to interconnect the cards. It is divided into five signal categories. The MultiBus II Parallel System Bus provides the primary communications among the cards and is a network based upon message/packet transmission. The Central Services Module Bus provides the network management signals for the MultiBus II Parallel System Bus. The Extension Bus provides unique signals (i.e., non-network type data) for the cards. Examples of these signals include timing clocks and frequencies. The Power Bus provides the secondary power signals, grounds, and appropriate reset signals. The Local Bus provides a simple, low overhead, network. The Local Bus is primarily for processors to operate with remote memory devices.

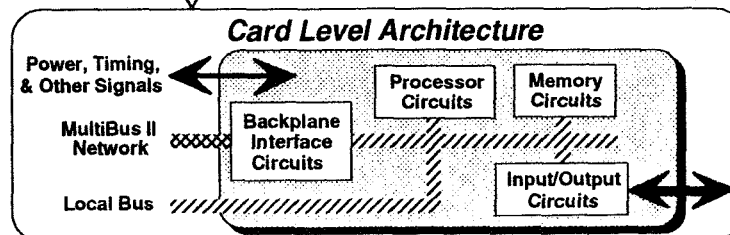
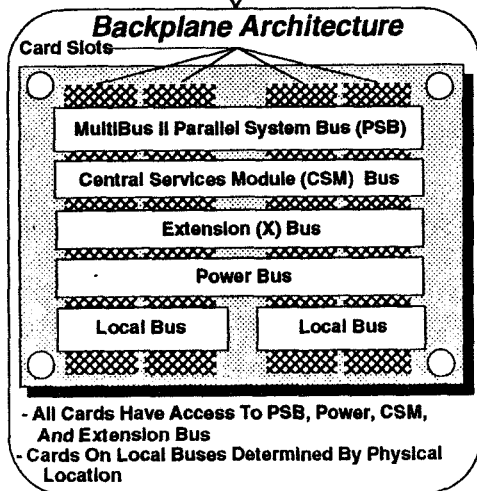
Each card, illustrated in the viewgraph, uses the same general architecture. The architecture is based on four components interconnected with a Local Bus. The Backplane Interface Circuits are for communications across the backplane using the MultiBus II network and for network management using the Central Services Module Bus. Processor Circuits and Memory Circuits provide, where required, the computing environment to host the functions allocated to the card. The Input/Output Circuits provide the signal conditioning and handshaking between the card and external devices. A Local Bus interconnects the on-board components together and may be extended into the backplane.

The box level architecture is decomposed into a software architecture in addition to the backplane and set of cards. From an abstract view, the software is allocated to two layers that are allocated to the various TCP cards. The Flight Systems Services CSCI software layer provides the transition from hardware devices to the user environment. This layer provides software to operate the hardware and to perform basic computing services (e.g., communications, tasking, timing, and file/data manipulations). In addition, modules reside in this layer to provide common programming interfaces for application software. The Flight Applications CSCI software layer hosts the application software and a flight executive manager that manages the application software.



**Hardware
Decomposition**

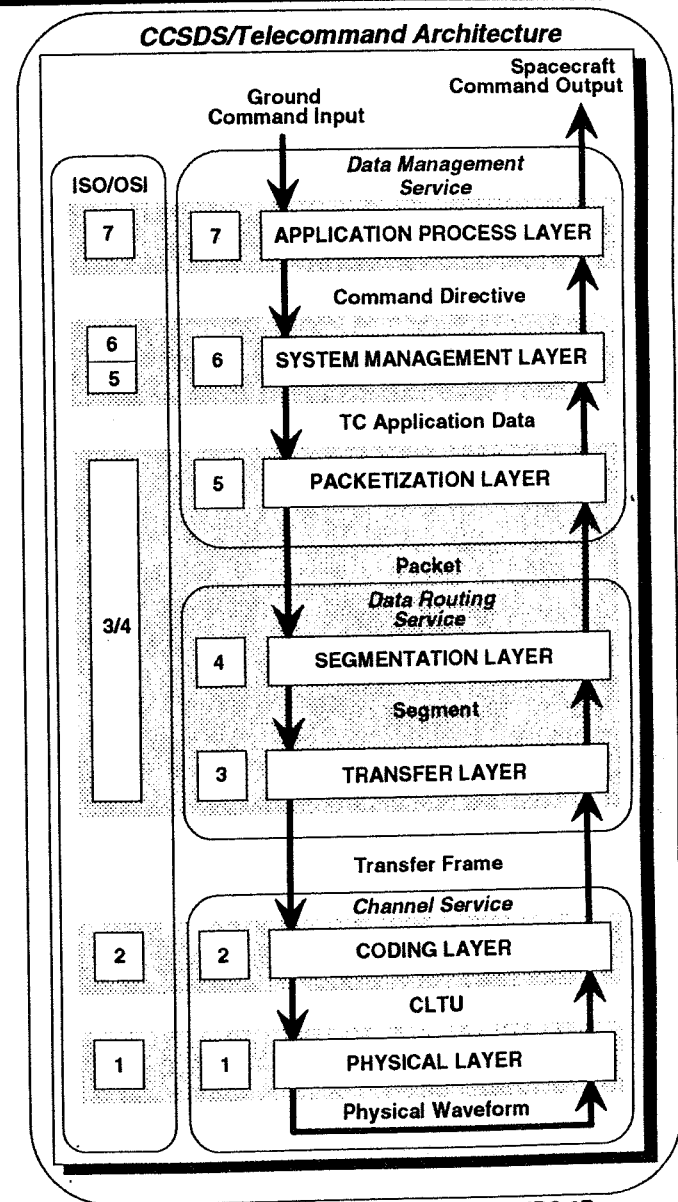
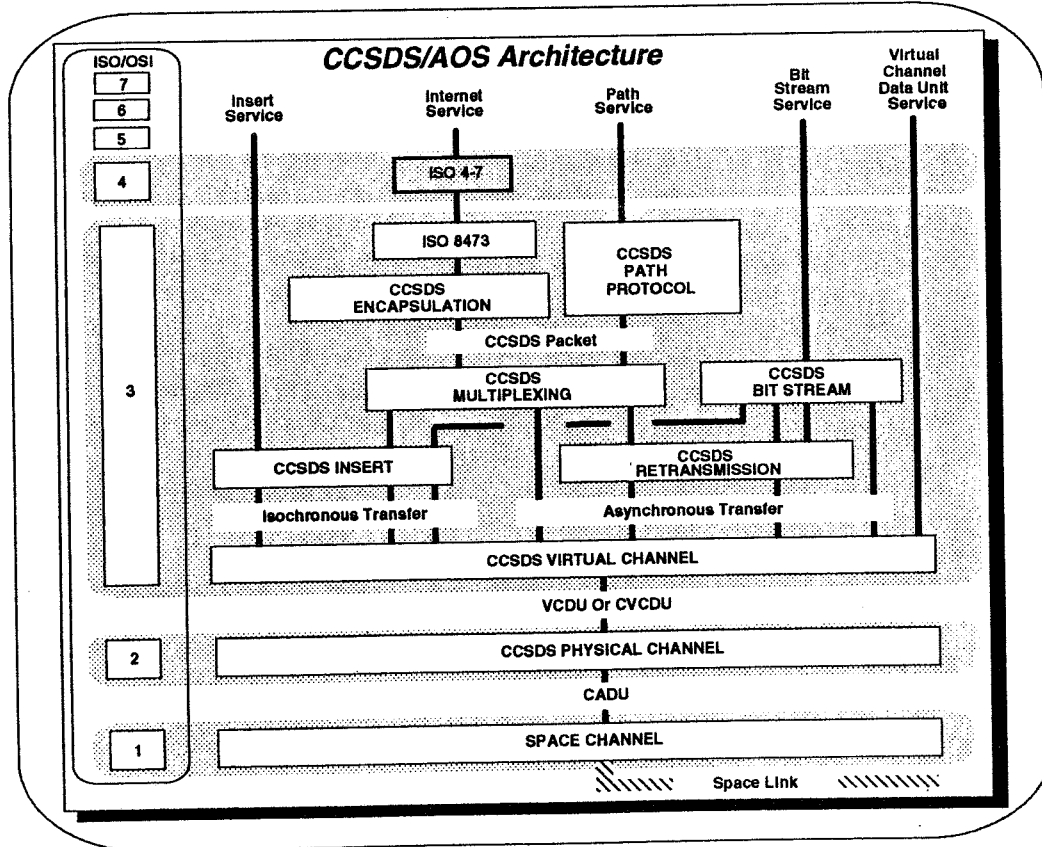
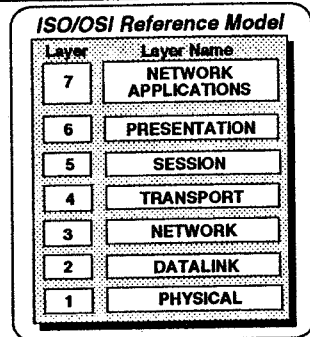
**Software
Decomposition**



Government, industry, and international standards and reference models are important considerations during development of control systems. Communication standards and models are important for spacecraft real-time systems like the TCP. In one sense, the TCP provides a "gateway" function between the spacecraft subsystems and the ground system. The TCP "gateway" performs protocol processing on the uplink/downlink, processes information, and performs protocol processing for the information exchange with the subsystems. In addition, the TCP provides peer-to-peer communications between the on-board subsystems and their corresponding ground subsystems.

The TCP uses a variety of networks to communicate with the spacecraft subsystems. Selection of each network is viewed with compatibility to the International Standards Organization/Open Systems Interconnect (ISO/OSI) Reference Model. The ISO/OSI model provides seven layers of services for efficient and reliable communications from one entity to another connected by networks. The underlying concepts of the model are to provide consistent and uniform interfaces between the layers and for each layer to provide higher levels of service than the layer underneath. The first layer, **Physical**, provides the hardware and media interconnections forming the network. The second layer, **Data Link**, moves information from one network device to another. It also provides flow and error control. The third layer, **Network**, provides additional services to move information segments and performs routing management. The fourth layer, **Transport**, provides reliable end-to-end data transfer between communicating users. The fifth layer, **Session**, establishes and manages connections between communicating users. The sixth layer, **Presentation**, provides data format translation to ensure that the data representation is understood by the communicating users. The seventh layer, **Applications**, provides basic data handling services for communicating users. Some of these services are Message Handling; File Transfer, Access, and Management; Virtual Terminal; Directory Services; and Network Management.

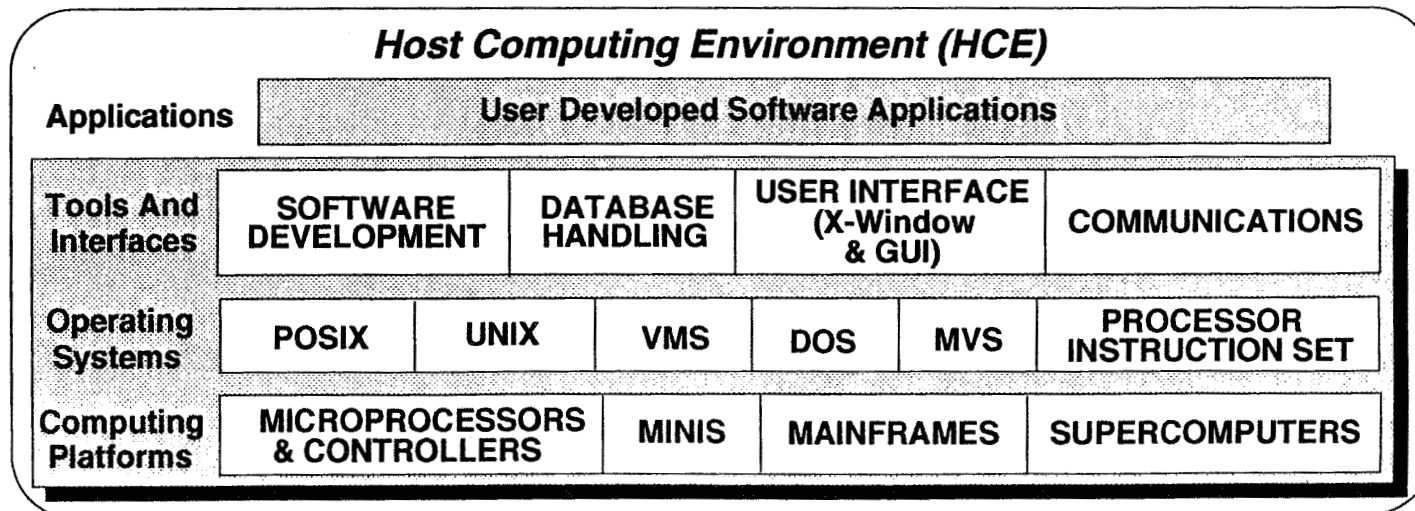
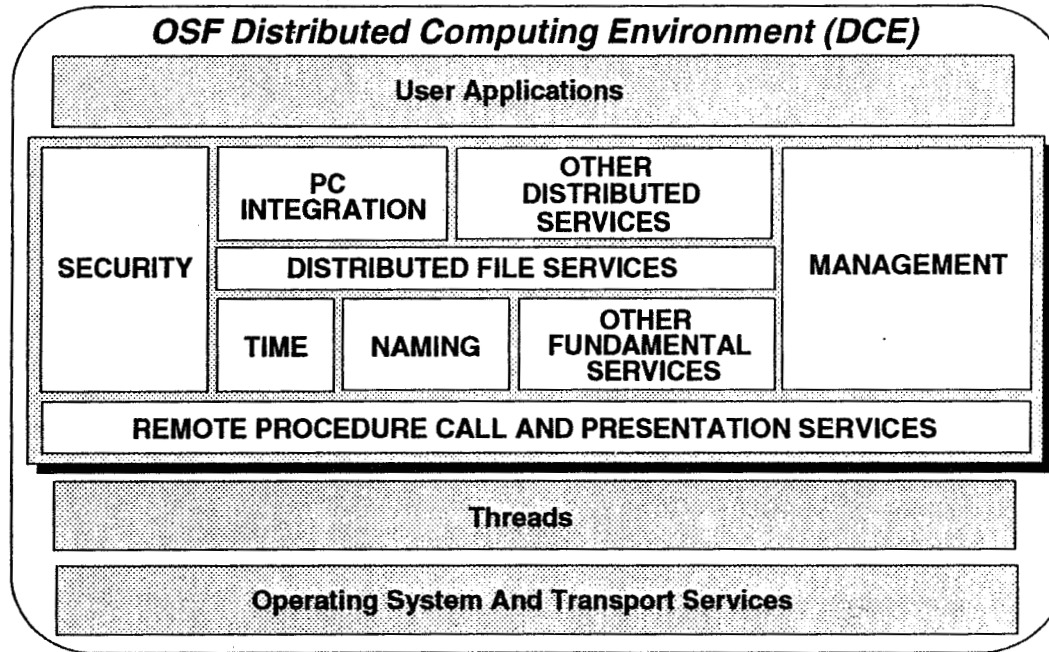
The recent emergence of the CCSDS protocols is of particular importance to the TCP. The primary objective of CCSDS is a new communication architecture, based on the ISO/OSI model, for communicating various data types between a spacecraft and the ground system. The Advanced Orbiting Systems (AOS) and Telecommand architectures of the CCSDS protocol suite are the basis of the TCP's Uplink and Downlink Cards. Telecommand provides reliable and efficient transfer of control information from a ground source to the spacecraft. Standardization of protocol layers and interfaces provides for a common means of ground-to-space communications. The AOS architecture is a full suite of data services between space and ground systems. Data types range from packets with well defined formats to bit streams that are unstructured. The architecture allows Internet and/or path data units to transfer across multiple interconnected subnetworks. AOS has been designed from packet and virtual channel technologies to provide a dynamic means to efficiently assign bandwidth on an as needed basis. Both Telecommand and AOS enhance re-use among multi-missions and cross-support among multiple ground resources/agencies.



Similar to communication standards and models, computing standards and concepts are important. As a "gateway", the TCP provides a computing environment to process commands for distribution to the subsystems and to process data for transmission to the ground. In addition, the TCP provides a general purpose computer to host flight application software. This software receives data from the subsystems, performs algorithm processing, provides processed data for the downlink, and generates control information for the subsystems. An objective of the TCP is for flight application software to operate in a peer-to-peer interaction with its corresponding ground software. To achieve this objective, open systems, modularity, layering, and distributed computing concepts are design drivers to the development of the TCP architecture.

General computing environments contain four overall layers in its architecture. Layer one, **Computing Platforms**, contains the computing hardware devices. These devices range from microprocessors/controllers to supercomputers. Layer two, **Operating Systems**, provides the basic software to manage the underlying hardware devices. This layer ranges from instruction sets for individual processors to full functional operating systems like UNIX, DOS, and VMS. The open system concepts have led to a new full functional operating system that provides common and consistent application programming interfaces independent of the underlying hardware. This system is the Portable Operating System Interface (POSIX). Layer three, **Tools And Interfaces**, provides basic data handling services for the user applications in addition to the software development and test environment. Some basic tools and interfaces include file manipulations, data base handling, user interfaces (e.g., graphical user interfaces, window managers, and display/keyboard controls), and the linkage to communication services that transfer data among applications and between applications and external hosts. Layer four, **Applications**, contains the user developed software applications that configure the computing environment to perform the intended mission.

The Open Software Foundation (OSF) has established a Distributed Computing Environment (DCE) reference architecture to promote interoperability within a heterogeneous, networked, computing environment. The primary goal of DCE is to provide a complete, integrated, and uniform set of services to support distributed applications regardless of the underlying hardware. The DCE architecture is layered with the bottom layer providing basic interaction services with the host platform. The highest layer interacts with the user applications. Threads support concurrent programming, multiple executions, and synchronization of global data. It is ideally suited to support client/server interactions. Presentation services provide translations to ensure that the data representation is common to the distributed users. Remote Procedure Call establishes the connection between communicating applications on different hosts. Time ensures a single time reference is used between applications on different hosts. Naming identifies distributed resources on the network. Distributed File Service implements the client/server model and enables global file accesses to appear as a local access. PC Integration allows minicomputers, mainframe, and PC users to share resources in a distributed environment. Security provides the distributed environment with authentication, authorization, and user account management services.





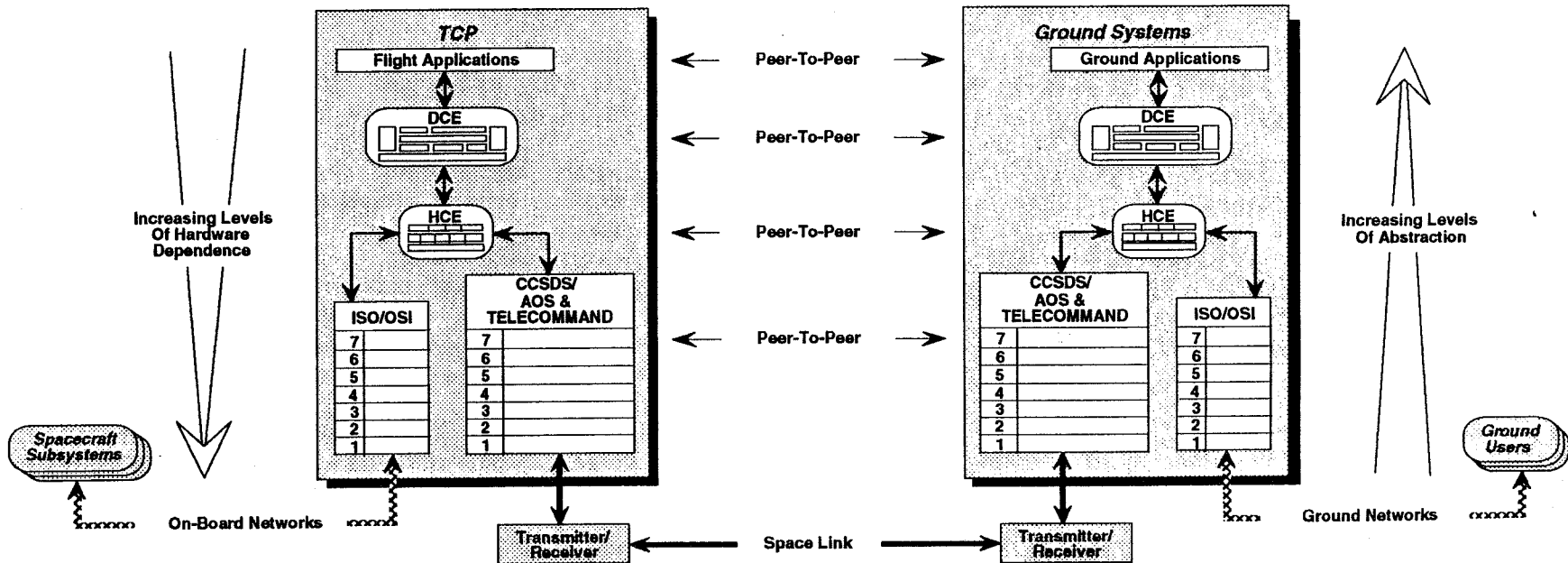
THE REFERENCE MODELS ARE INTER-RELATED WITHIN THE TCP TO FORM PEER-TO-PEER INTERACTIONS WITH THE GROUND

The emergence of many models and reference architectures requires a system architect to understand the interactions and relationships of the models when building a new system. Each of the models describes an important aspect of networking and distributed processing and their role in supporting open systems and peer-to-peer communications. The CCSDS, ISO/OSI, Host Computing Environment (HCE), and DCE models are relevant to the TCP. The attached viewgraph depicts the overall model relationships within the TCP and its peer-to-peer communications with the ground system.

The end-to-end system is viewed as three networks connected together to form the path between the end user(s) and the spacecraft subsystems (e.g., sensors and actuators). The first network connects the end user(s) with the ground system. It may use any available standard network (e.g., Ethernet, Internet, or X.25) or may be custom. The second network, space link, connects the ground system to the spacecraft/TCP. The third network connects the TCP to the end sensors/actuators. It may also use standard networks or may be custom. Both the ground system and TCP provide "gateway" functions in the sense that both form the linkage between two different networks and perform the protocol processing for each connected network. For the space link, the ground system and TCP use the CCSDS AOS and Telecommand architectures. The physical path is established at layer 1 with peer-to-peer communications at the upper layers. For the ground network and on-board networks connecting to their respective users, both employ protocol processing based upon the ISO/OSI model.

In addition to the "gateway" functions, the ground system and TCP provide important processing capabilities to manage the spacecraft mission. This processing consists of the HCE, DCE, and applications software. The HCE provides the general computational platform for hosting the applications in addition to providing the connection with the networks. The applications software would reside directly on the HCE if it were not desirable to support distributed processing between the TCP and other on-board subsystems or between the TCP and the ground system. However, because distributed processing provides significant advantages to systems such as easier global access to data and higher levels of abstraction to the user, the DCE is placed between the HCE and the applications.

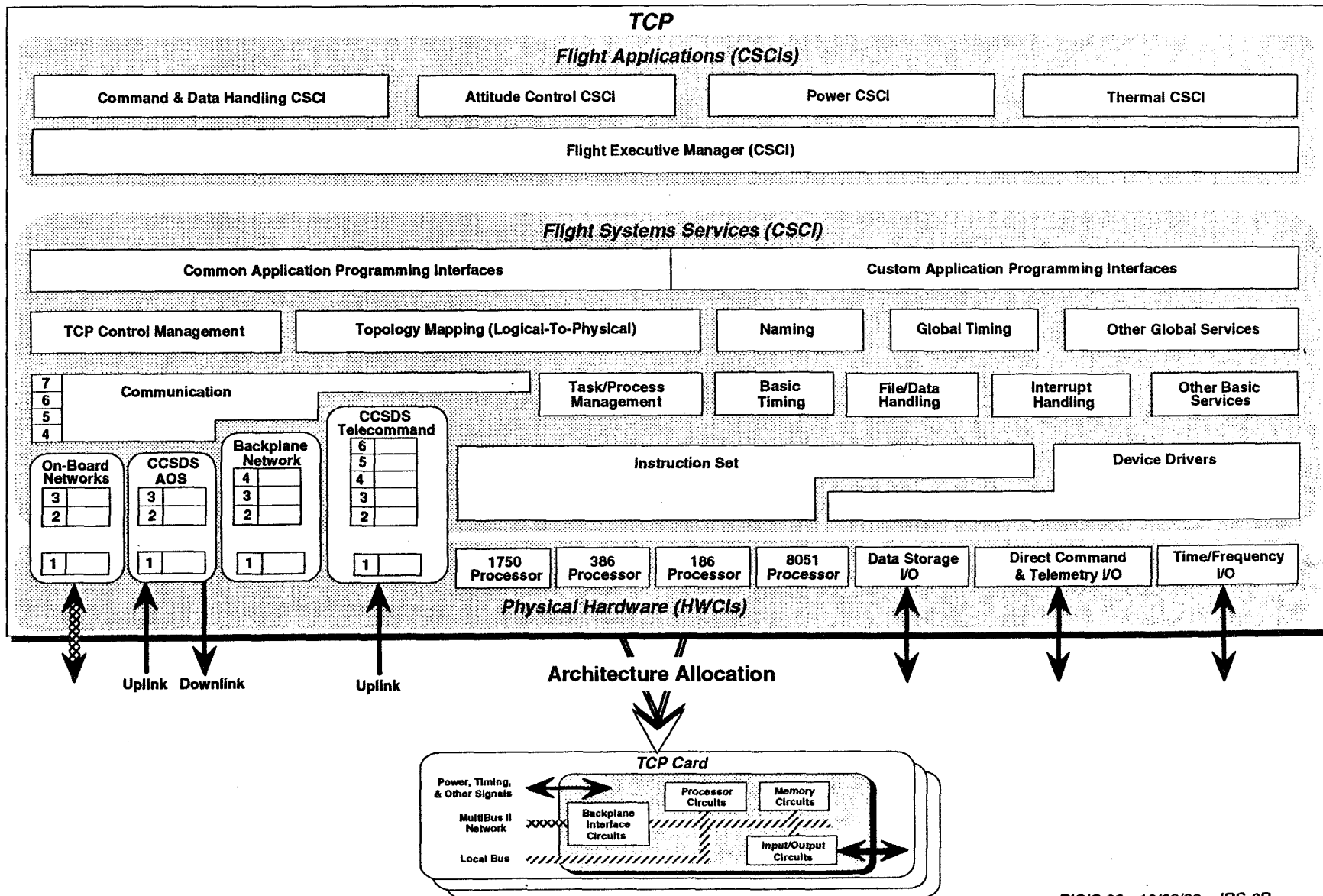
The combination of models forms an architecture that promotes hardware independence and abstraction. At the lowest levels, the architecture is highly dependent upon the hardware implementation and its resident operating system. User applications at this level must know them in detail. As user applications are moved higher up in the architecture, the underlying hardware and configuration become hidden from them and the underlying layers provide higher and higher levels of services. At the highest level, the application-to-system interface becomes purely logical where the application need specify only what it needs. The system will perform the implementation of the need.



The CCSDS, ISO/OSI, HCE, and DCE models are important inputs to the TCP architecture development. However, two negative aspects of these models are 1) they contain a significant amount of processing overhead and 2) they provide many services not required for a particular implementation. Generally for spacecraft missions, the control systems are constrained by size, power, and weight requirements. These constraints limit the amount of processing capability that can be achieved. Environmental factors (e.g., radiation, shock, vibration, and thermal) also are factors in the amount of available processing capability. In addition, user applications are the highest priority for mission success with the housekeeping functions being the lowest priority. Within the constraints, a prudent "stripping down" of the models can be achieved while still maintaining the overall concepts of open systems, distributed processing, and networking. The general technique used within the TCP is to maintain the model's lower layers intact and replace the upper layers with a single, efficient, software module that preserves the outer interfaces and as much of the services as possible. In addition, the layers and services not required for flight are removed. As the computing performance/size ratio improves through advancements in processor technology, new higher performing processors can be inserted into the TCP allowing for the addition of those layers and services that were initially removed.

The attached viewgraph shows the TCP architecture. It illustrates the use of the models and modular and layered design techniques. The architecture consists of three high level layers (Physical Hardware, Flight Systems Services, and Flight Applications) that contain sub-layers. The Physical Hardware layer contains all the hardware devices including processors, dedicated I/O devices, and communications media contained within layer 1 of the networking models. The Flight Systems Services layer contains sub-layers using elements from the ISO/OSI, CCSDS, HCE, and, DCE models. In the case of communications, the protocol stack is maintained to at least layer 4 and, for Telecommand, to layer 7. The remaining upper layers have been combined into one software module. It serves the various networks and provides the linkage to the HCE. For the HCE, a full operating system is not used. Rather, the processor instruction sets are used coupled with programs to perform task/process management, basic timing, file/data handling, and interrupt handling. The next sub-layers use the DCE model to provide higher levels of abstraction and hardware independence to the flight applications. The provided services include the overall management of the TCP, naming, global timing and synchronization, resource mapping that translates from logical names to physical locations as one of its features, and common programming interfaces. Within the flight applications layer is a flight executive manager that provides management of the upper mission application programs like attitude control, power, and thermal.

The overall TCP architecture is allocated to the individual TCP cards and, subsequently, to the individual major components on the card. For example, the backplane network protocol stack is allocated to the backplane interface circuits for all cards. For the downlink card, the CCSDS AOS network is allocated to the input/output circuits. And for the on-board computer card, almost all of the flight system services and flight applications layers are allocated to the processor circuits.





TCP - DESIGNED AND BUILT FOR THE NEXT GENERATION SPACECRAFT

In summary, this symposium describes Mission Safety Critical Systems that have high criticality to the successful operation of a mission. The TCP, a computer based real-time control subsystem, is one of them. The TCP is the foundation of Command & Data Handling Subsystems. It provides command handling, spacecraft health control and monitor, time management, data storage, data exchange with the ground, and the hosting of flight applications software. Its architecture is based upon communication and computing reference models and architectures. Modularity and standardized interface concepts have led to a TCP composed of a set of modular cards connected together through a backplane with the cards containing layered software. The architecture enhances optimum re-use and transportability to support rapid mission adaptability while reducing costs and shortening development schedules.



**TCP - DESIGNED AND BUILT FOR THE
NEXT GENERATION SPACECRAFT**

In Summary, The TCP

- * Is An Important Mission & Safety System For The Success Of The Spacecraft Mission**
- * Provides Command Handling, Spacecraft Health Control And Monitor, Time Management, On-Board Processing, Data Storage, And Data Exchange With Mission Personnel**
- * Uses ISO/OSI Reference Models, CCSDS Protocols, Open System And Distributed Processing Concepts, And Packet Techniques**
- * Is A Set Of Modular Cards With Layered Software That Enhances Re-Use And Transportability To Support Rapid Adaptation To Individual Missions Thereby Reducing Costs And Shortening Schedules**

Session III

REAL-TIME COMMUNICATIONS

9:00-10:30

Session Leader:

Wei Zhao, Texas A & M University

Presenters:

Scheduling Real-Time Traffic with Dual Link Networks

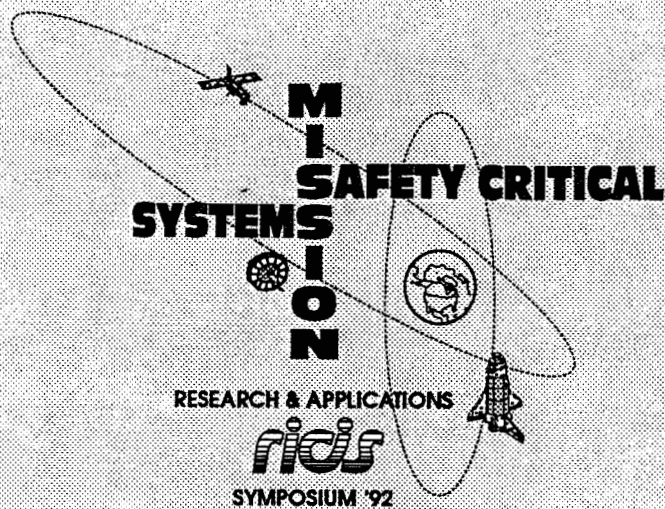
**Lui Sha, Shirish Sathaye & Jay Strosnider, *Software Engineering
Institute, Carnegie Mellon University***

**CAPTIONALS: A Computer Aided Testing Environment for the
Verification and Validation of Communication Protocols**

**C. Feng, X. Sun, Y.N. Shen & F. Lombardi
*Texas A & M University***

**Performance Comparison of Token Ring Networks
for Real-Time Applications**

**Sanjay Kamat & Wei Zhao
*Texas A & M University***



REAL-TIME COMMUNICATIONS

Session Leader: Wei Zhao

BIOGRAPHY

Wei Zhao received the Diploma in physics from Shaanxi Normal University, Xian China, in 1977, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, in 1983 and 1986, respectively. He is currently an Associate Professor in the Department of Computer Science, Texas A & M University.

He has published extensively in the areas of scheduling algorithms, communications protocols, distributed real-time systems, concurrence control in database systems, and resource management in operating systems. He received the Best Paper Award in the IEEE International Conference on Distributed Computing Systems for a paper on hard real-time communications. He was a Guest Editor for a special issue of ACM Operating System Review on Real-Time Operating Systems. Currently, he is an editor for the IEEE Transactions on Computers. He will Co-chair the IEEE Workshop on Imprecise and Approximate Computation, to be held in December 1992.

Scheduling Real-Time Traffic on Dual Link Networks

Lui Sha

Shirish S. Sathaye

Jay K. Strosnider

Carnegie Mellon University

Pittsburgh, PA 15213

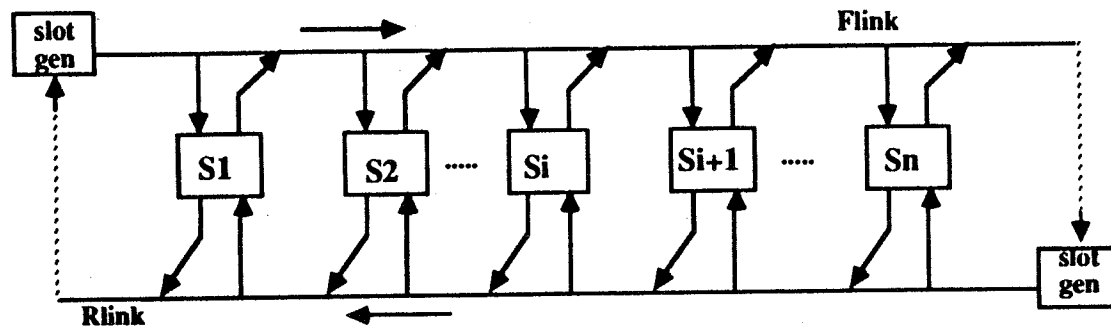


Outline

- Dual-Link Network Architecture
- Priority driven Scheduling Support
- Sources of Unpredictability
- Network Coherence
- Dual-Link Network Scheduling Results

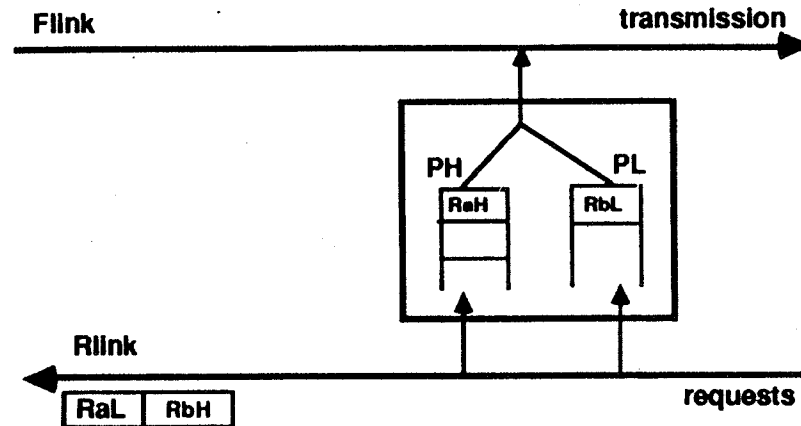
More details in: *Analysis of Reservation-Based Dual-Link Networks for Real-Time Applications*, CMU/SEI-92-TR-10

Dual Link Architecture



- Fixed size slots on each link
- Slots on link reserved by making requests on opposite link.
- Stations count requests and let one empty slot go by per downstream request.
- A station can make a new request at same priority when original request is satisfied.

Priority Based Reservation



Let a station have a slot to transmit at a certain priority:

- It sets a REQ bit at appropriate priority and enter the request into the queue.
- If an empty slot passes on the Flink the station dequeues the highest priority request.
- If the dequeued request is a self-entry the station uses the slot.

IEEE 802.6 DQDB - A Dual Link Implementation

- IEEE 802.6 Distributed Queue Dual Bus: Standard Metropolitan Area Network.
- To be deployed as public/private network.
- Slot size and format compatible with ATM/BISDN.
- Expected to carry voice, video and data traffic.
- Real-time version to carry plant control traffic in addition to voice, video, data.



Scheduling Dual Link Networks

CONNECTION: A periodic message stream $\tau_i : (C_i, T_i)$ from a source in the network such that it generates C_i packets every period T_i

Transmission Schedulability:

A connection is transmission schedulable, if after initial startup delay, it can transmit C packets every period T

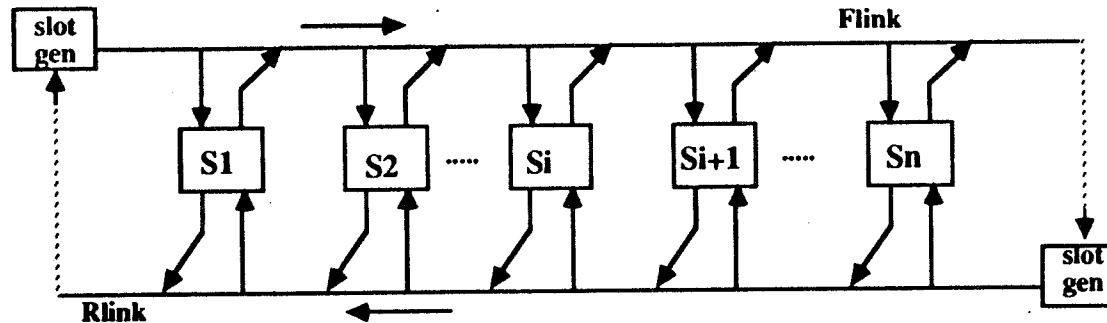
End-End Delay = Initial Delay + T + Prop. Delay

Priority Driven Scheduling Support

The good news:

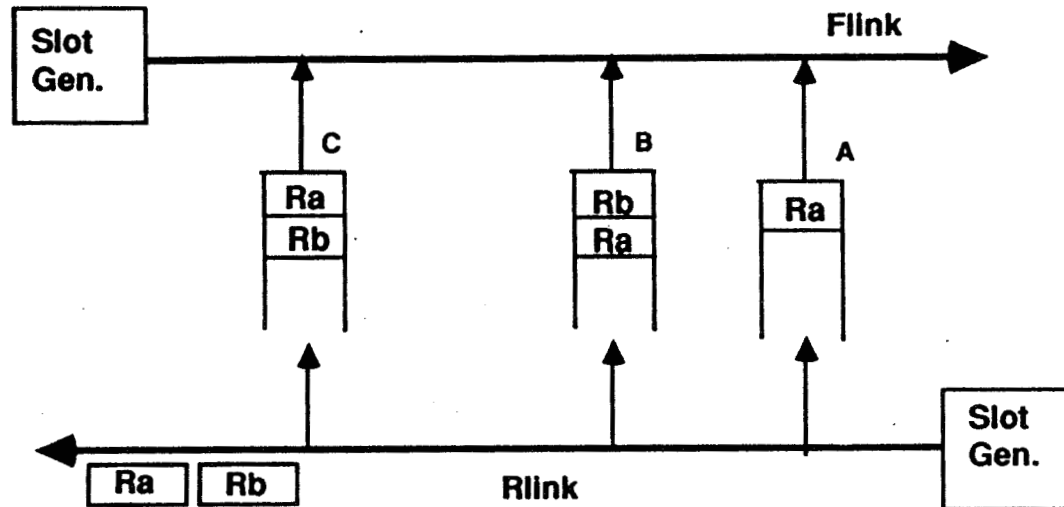
- Requests can be made at different priorities.
- Stations have priority request queues.
- Station maintain a global queue to capture network state.
- Stations do not have to wait for "token rotation" as in some token passing protocols.
- Concurrent transmission and arbitration.
- Short packets, create more preemption points.

Difficulties in Implementing Priority Driven Scheduling



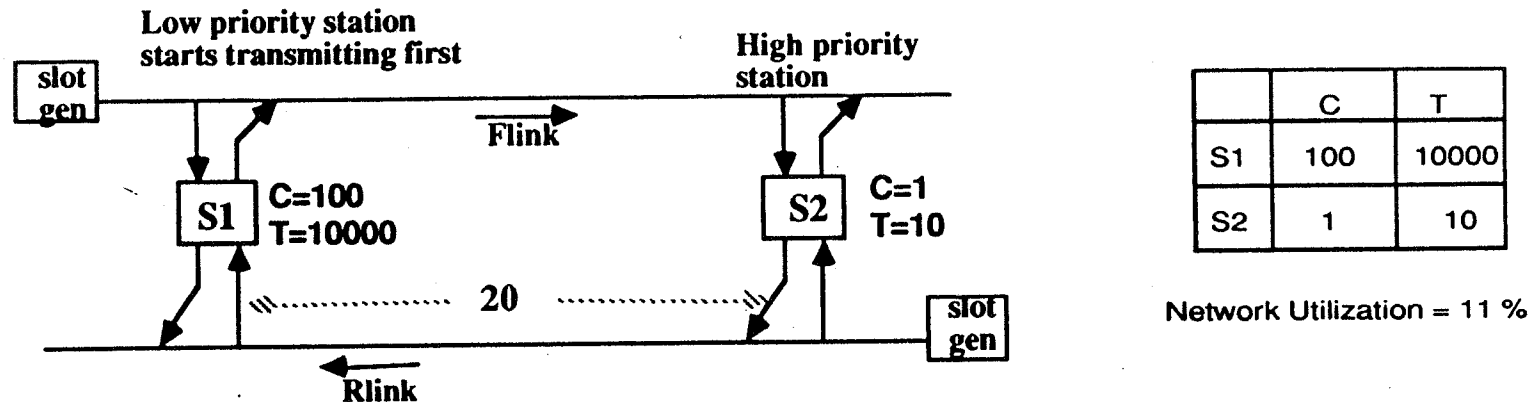
- Network Scheduling inherently different from Centralized Resource Scheduling
- Incomplete and/or delayed information, depending on distance on distance and relative position.
 - Request from S_i observed after a delay by S_2
 - Request from S_2 never observed by S_i
- Achieving predictable operation is challenging.

Inconsistent Queues



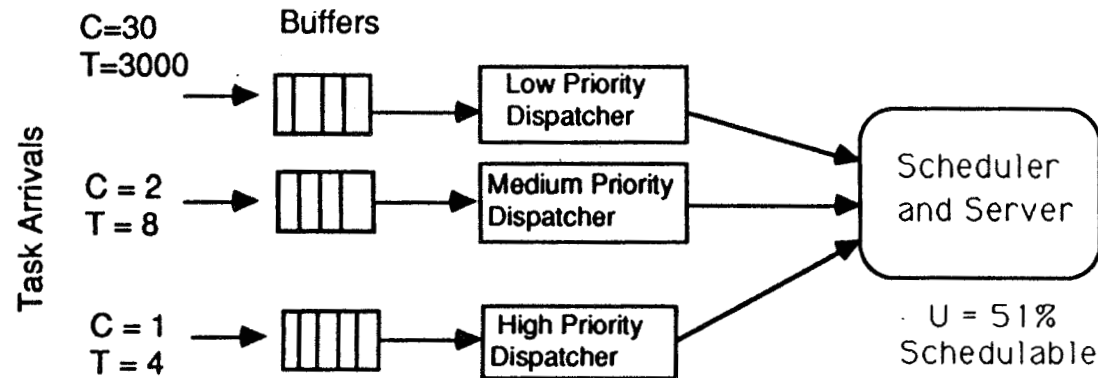
- Station A makes self-entry and request Ra on Rlink.
- Station B makes self-entry, but observes Ra before it can make Rlink request. Enters Rlink queue. Makes Rlink request.
- Station C observes Ra and Rb on Rlink and enters them in its queue in observed order.

Unpredictability due to Request Throttling observed in IEEE802.6



- Station cannot have multiple unfulfilled requests at a priority.
- Let S2 make a request on Rlink
- Then let S2 observe experiences 100 consecutive busy slots on Flink, preventing it from making additional requests.
- S2 cannot make 1 request every 10 slots, hence cannot transmit once every 10 slots.
- S2 not t-schedulable even though network load is 11 %.

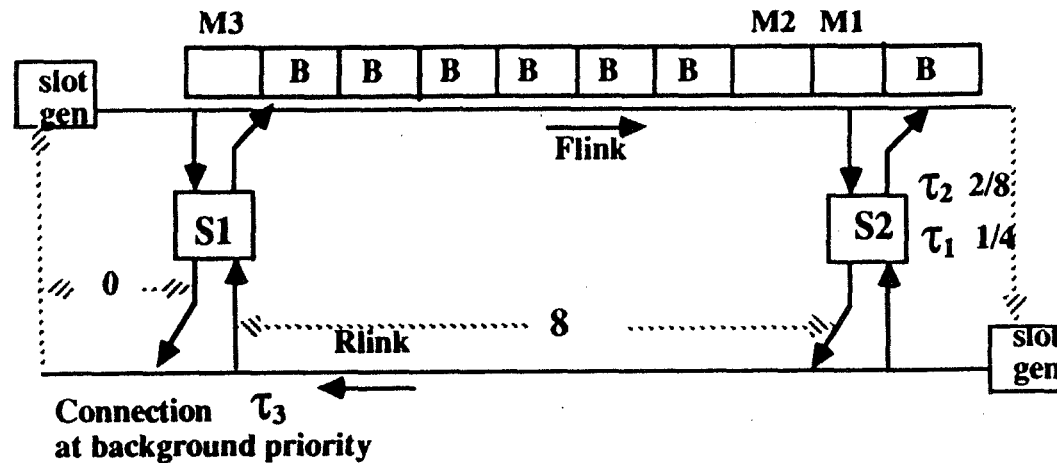
Sleeping Dispatcher Effects



Consider above centralized system:

- In steady state, low and medium priority jobs arrive periodically, are dispatched and scheduled, meeting deadlines.
- Let high priority tasks begin to arrive periodically. Dispatcher is "sleeping". They accumulate in buffer (deadlines missed)
- Suppose high-priority dispatcher "wakes up". Low and medium priority jobs are preempted by multiple high priority jobs.
- Low and medium priority jobs miss deadlines. High priority deadlines already missed.

Example : Effects of Over-preemption



Conn.	Packets	Period	Priority
τ_3	30	3000	B
τ_2	2	8	M
τ_1	1	4	H

Network Utilization = 51 %

- Background and medium priority message streams schedulable in a steady state.
- At time t_0 , high priority message stream set up at S2. It make 1 request on Rlink every 4 slots (no request throttling).
- S2 observes busy Flink slots. High priority requests build up in S2's transmission queue.
- High priority message uses M1 and M2 which would have been used by medium priority message
- Both high and medium priority message streams miss deadlines.

Coherence: A Foundation for Predictability

- Intuitively,

Coherence: a logical and orderly relationship between elements of a system.

- In a coherent system we can reason about relationship between requests on Rlink and slot usage on Flink.
- Properties of a Coherent Dual Link Network:
 - Lossless Station Queues
 - Consistent System: Ordering of Requests in different queues must be consistent
 - Bounded Priority inversion

Achieving Network Coherence

Lossless Station Queues

- Station logic must be fast enough

Consistent System

- Stations make own request on Rlink prior to queue entry.
- Higher priority to local requests than equal assigned priority requests on Rlink.

Bounded Priority inversion

- Priority Station Queues.
- Rlink requests independent of Flink traffic.
- Preemption of low priority requests. Preempted requests placed in outgoing req. queue in priority order.

But Coherence by itself does not result in predictable system, due to the effects of *Over-preemption*.

Media Access Protocols

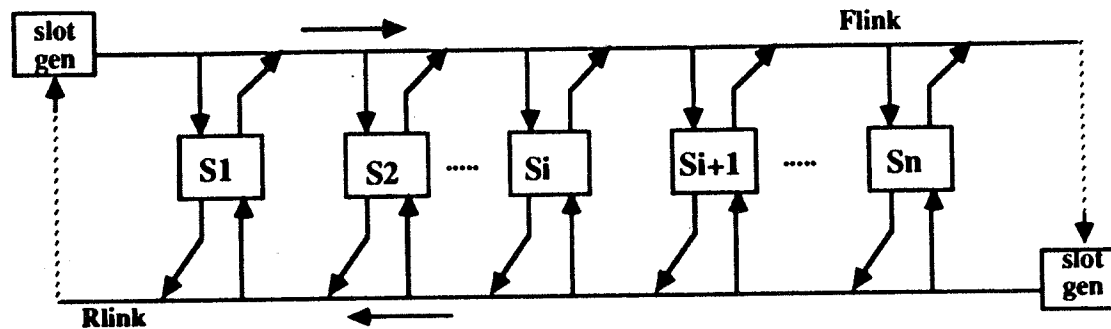
Coherent Reservation Protocol: Achieves

- Lossless Station Queues
- Consistent Station Queues
- Bounded Priority Inversion

Preemption Control Protocol:

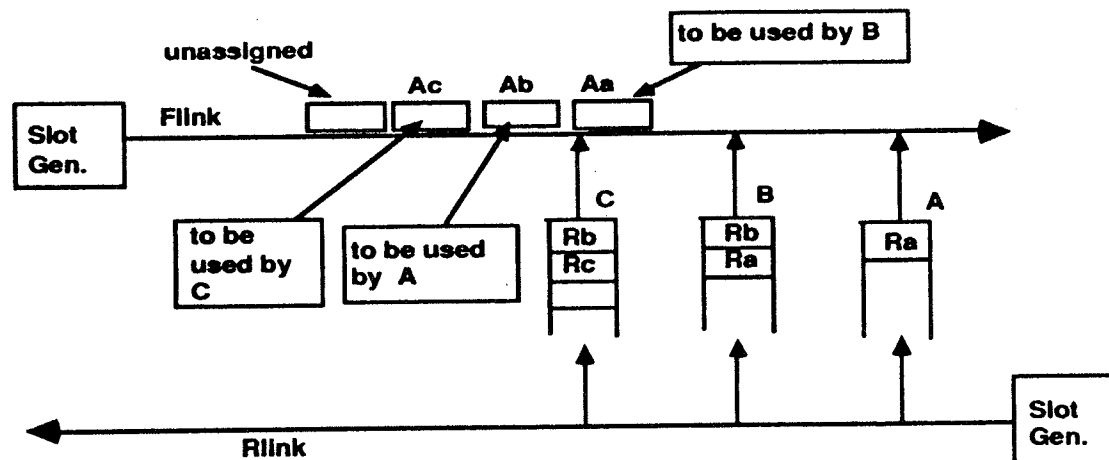
- New connections make requests every period.
- Wait for a period of $2d_i$ before using slots on Flink.
- Use no more than C empty slots every period T on Flink.

Virtual Slot Assignment



- **Imagine** that slot release by link slot generator immediately after a receiving a request from Rlink is **assigned** to the station that made request.
- Virtual assignment: An **abstraction** to reason about relation between request patterns on Rlink and slot usage pattern on Flink.
- If every station can use its virtually assigned slot we may be able to say something about predictable operation.

Unpredictability of Inconsistent Queues



- Unpredictable behavior depending on position of a virtually unassigned slot.
- Unpredictability can be avoided if queues are consistent.

Analysis of Coherent Dual-Link Networks

Theorem: In a dual link network that follows CRP, station queues are consistent with each other.

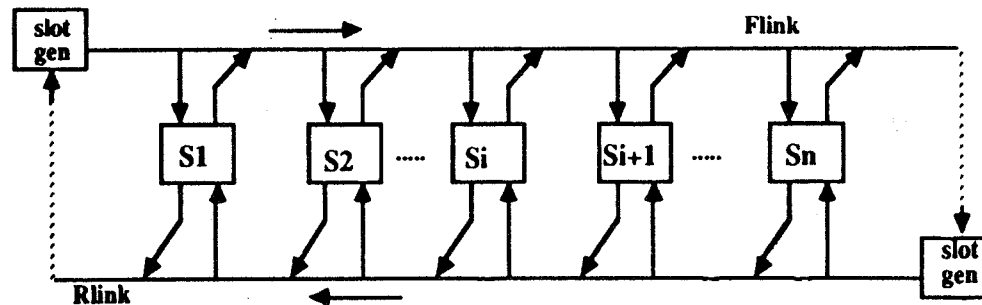
Theorem: In a multi-priority coherent system a request cannot be satisfied by a slot that is virtually assigned to a higher priority or higher effective priority station.

Theorem: For any periodic connection in a coherent dual link network, the maximum duration of priority inversion is bounded by $2d_i$, where d_i is the distance in slot times between the source station and the Flink slot generator.

Critical Instant Phasing Time Space Equivalence

Critical Instant Lemma in Centralized System Scheduling:

Given a set of periodic activities in a centralized system, the longest completion time for any activity occurs when it is initiated at the *critical instant*. The critical instant is the time at which the activity is initiated along with all tasks of higher priority.



Lemma: Given a set of periodic connections in a dual link network, the longest delay experienced by any request initiated at $t=0$ from a station, is no greater than the delay that would have resulted if all higher priority connections were located in the same station and generated a request at $t=0$.

Schedulability Results

Theorem: Given a set of periodic connections, if the set of connections is schedulable in a centralized, preemptive, priority driven system, then the set of connections is t -schedulable in a coherent dual link network.

Theorem: In a t -schedulable coherent dual-link network, a connection with C packets to transmit every period, will require $\lceil 2d_i/T \rceil$ buffers in the source station of the connection.

Revisiting our Paradigms

Preemption:

Over-preemption can be harmful to predictable behavior.

Schedulability:

T-schedulability: A separation of resource contention delay and propagation delay is a more meaningful measure of wide-area network schedulability.

Critical Instant Phasing:

There is a space (distance) equivalence to critical instant phasing.

Tie-Breaking between Equal Priority Requests:

Local requests have higher effective priority

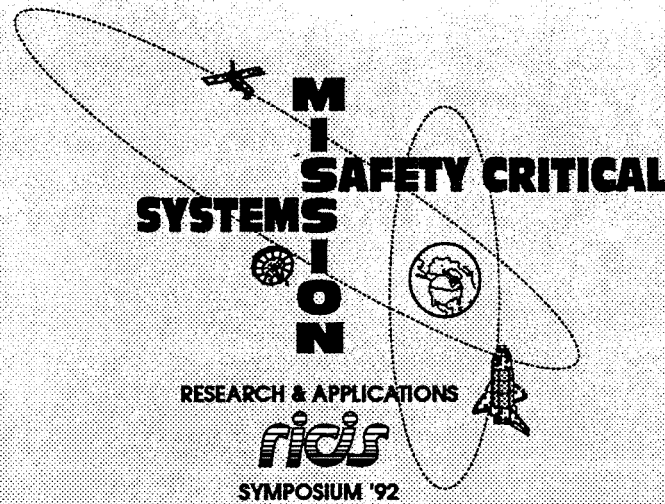
Concluding Remarks

The Problem we faced:

Network scheduling inherently different from centralized system scheduling; distributed decisions with incomplete information.

Solutions:

- Extended real-time scheduling theory to accommodate distributed nature of the system.
- Introduced the concept of coherence as a foundation to achieve predictability in a dual link network.
- Demonstrated that a coherent dual link network can be analyzed as though it were a centralized system.

27237
P. 22

1995107749

504058

22P.

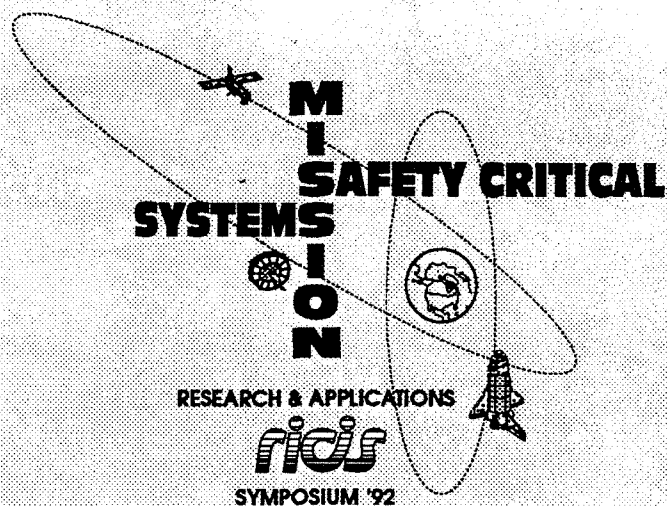
CAPTIONALS: A COMPUTER AIDED TESTING ENVIRONMENT FOR THE VERIFICATION AND VALIDATION OF COMMUNICATION PROTOCOLS

C. Feng, X. Sun, Y.N. Shen and F. Lombardi

ABSTRACT

This talk presents novel issues involved in the verification and validation of protocols for distributed computer and communication systems using a computer aided testing approach (CAT). Verification is the process which substantiates the accuracy of the specifications of a protocol; validation is the process in which the validity of the specifications to meet the desired objectives (or requirements) is confirmed. They make up the so-called process of conformance testing. Protocol implementations which pass conformance testing, are then checked whether they can operate together. This scenario is referred to as interoperability testing.

A new comprehensive approach to protocol testing is presented. This approach addresses new fundamental issues for testing protocols: (1) modeling for inter-layer representation for compatibility between conformance and interoperability testing. (2) computational improvement to current testing methods by using the proposed model inclusive of formulation of new qualitative and quantitative measures (such as detectability) and time-dependent behavior. (3) analysis and evaluation of protocol behavior for interactive testing without an extensive use of simulation. These problems careful require definition and are analyzed using the proposed CAT approach: (1) modeling of protocol activities at different levels of abstraction (inter-layer) to facilitate a cohesive approach to both conformance and interoperability testing through the use of a new analytical model (based on cellular automata); (2) design of a set of tools for interactive



testing by analytical techniques with partial reliance on simulation. (3) evaluation of new qualitative and quantitative measures for protocol testing through the development and use of appropriate interactive analytical tools as well as a testbed evaluation. The applicability of the proposed approach to real-life protocols (such as the abort sequence for the Space Shuttle) will be presented.

FABRIZIO LOMBARDI

BIOGRAPHY

Fabrizio Lombardi was born in Formia (Italy). He graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electric Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received the Master in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978) and the Ph.D. (1982). He is currently an Associate Professor in the Department of Computer Science at Texas A & M University. He was the recipient of the 1985/86 Research Initiation Award from the IEEE/Engineering Foundation. In 1988, he was awarded the Visiting Fellowship at the British Columbia Advanced System Institute, University of Victoria. Dr. Lombardi was also a co-director of the NATO Advanced Study Institute on Testing and Diagnosis of VLSI and ULSI, June 1987, Como (Italy). His research interests are verification and validation of protocols, fault tolerant computing, VLSI testing, and real-time systems. Dr. Lombardi is a Distinguished Visitor of the IEEE Computer Society and a Research Fellow of the Texas Engineering Experiment Station. Dr. Lombardi is the Program Chair of the 1992 IEEE International Workshop on

The Computer Science Department

**CAPTIONALS: A Computer Aided Testing
Environment for the Verification and Validation
of Communication Protocols**

By

C. Feng, X. Sun, Y.N. Shen

and

F. Lombardi

**Texas A&M University
Department of Computer Science
College Station**

Tel: (409)845-5464

Texas A&M University

Environment

Computer

Aided

Protocol

Testing by

Input

Output Behavior and By

Numeric

And

Logic

Specifications

- Deterministic Behavior

Protocols

- Complex entities/combination of hardware and software with no perceived separation of the two.
- Black box characterization
- Operation specified by international standards
- Interoperability among multiple protocol entities.

Important Issues

- Incompleteness in specifications
- Ambiguities in behavior due to inherent complexity
- Limited controllability and observability in the sequencing of the operation of the protocol entity
- Control the state explosion phenomena for representing a protocol (or combination of protocols)
- Ease of use for interactive operation

Previous Environments

- Postman by AT&T Bell Labs :
(Graph Based Approach)
- Estelle:
(Semantic Based Approach)
- OSI/ISO convention:
(SDL Approach)
- Main disadvantages: restricted applicability; no generation of test sequence, only proof of existence of particular properties of a protocol; complex operation, not suitable to non-expert user; heavy computational requirements.

Major Objectives

- Verify and validate hardware/software computing/communication systems
- Generate sequence for conformance testing to specifications
- Specification identification using logic and numeric semantics
- Identify and eliminate ambiguities in specifications and guarantee safety
- Evaluate fault coverage and identify components which yield loss of coverage
- Testbed evaluation

Computer Aided Testing

- Set of (semi-automatic) tools provided to a non-initiated user
- On-Line (interactive) evaluation of different testing strategies
- Experimental validation by testbed (yet to be implemented using NSF funded SI grant)
- Conformance to international standards (for OSI through ESTELLE)

Organization

- **Modeling**

Protocol specified in terms of basic steps (atomic actions).

Atomic actions performed by abstract machines.

Each abstract machine defined with respect to a single attribute.

Machines combined through a dependability net to represent overall behavior through a multi-level representation.
(hierarchical representation)

Advantages: it separates requirements from specifications, reduces computational overhead due to simulation, allows consistency checking.

Organization (Continued)

- **Evaluation**

Generate using analytical techniques (on-line) figures of merit for test sequence evaluation and generation.

Performed using the trace of an abstract machine, or the dependability net of the automata.

Organization (Continued)

- **Tools**

Provide data model to capture protocol entities and their relationships (identification of operational features, composition and instantiation to define higher level of abstraction, generalization to support certification procedures).

Translate attributes as evolution of structures using dynamic algebra semantics.

Numeric Specifications

- Applicable to dependency due to sensor data and timing specifications.
- Discrete (bounded) range validated only at critical values.
- Reachability analysis for establishing critical paths in protocol execution.
- Reference to a master control must be provided to handle consistency.

Logic Specifications

- Interdependences in stimula (inputs) on the behavior of the protocol.
- Sequentiality of protocol execution broken at single specification to allow manipulation of logic expressions (as enabler) through minimization.
- Decomposition of the protocol by isolating logic (correlated) behavior.
- Minimization using traditional VLSI-CAD tools (e.g., Quine-McCluskey)

Example: Conditional Time-out

Assume a periodic task over 100 cycles (clock incremented on a single cycle basis).

Specifications:

Keep the green light on and the red light off provided either the green switch is not pressed during the first 50 cycles or the yellow switch is not pressed during the second 50 cycles else, turn off the green light and on the red light if a switch is pressed and ring a bell.

To turn off the red light and turn on the green press the yellow switch and ring a bell during the first 50 cycles.

Reset capability is provided.

Example: Conditional Time-out Protocol Representation

Two States:

S_1 = Green light on, red light off

S_2 = Green light off, red light on

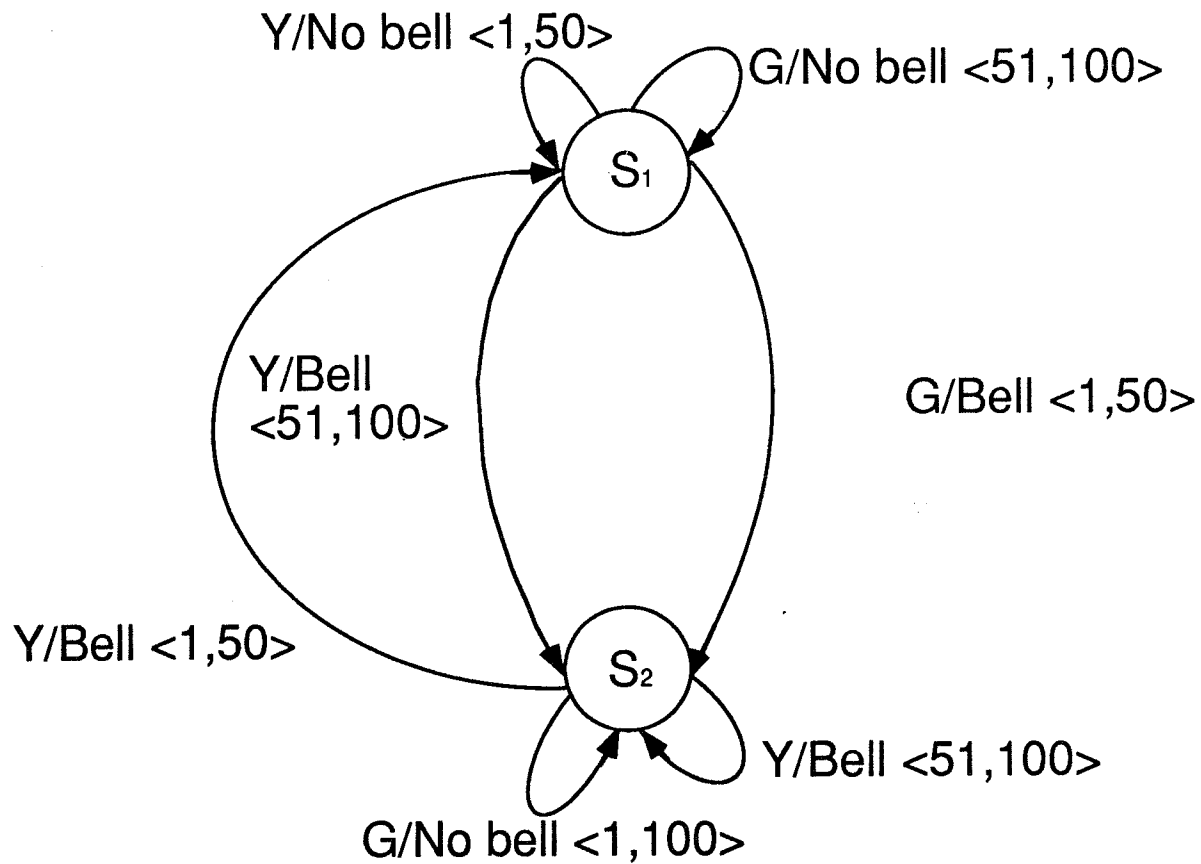
Output: Bell (or no bell)

Input: Yellow switch (Y),
Green switch (G)

Label: Input/Output

Clock: counter from 1 to 100 (and back)

Graph Representation



T = $\langle 1, 100 \rangle$

Reset edges not shown

Test Approach

- Touring: tour every edge of the representation at least once.
- Suite: generate a test subsequence for each type of fault which is allowed in fault model.
- Traverse an edge at lower and upper timing specifications using unique input/output qualifiers for the transitions (actions).
- Minimality achieved using a minimum spanning tree of the underlying time graph (by splitting each edge into multiple parallel edges to satisfy timing constraints).

Features

- Functional fault model (faults in logic and timing specifications)
- Test sequence generation as functions of inputs and time interval
- Ambiguity identification: press both switches (safe behavior due to single transition in protocol representation)
- Critical timing: 50th and 51st clock cycles
- Safety: both lights on (or off)

Evaluation

- Check every specification on both timing bounds.
- For interoperability: assume two equal protocol entities (X, Y), X as initializing the process.
- Bell output of first protocol (X) corresponds to green switch on of the other protocol (Y) and vice versa (yellow switch still a primary input).
- Assume only one protocol implementation to be faulty; then full controllability is lost (lost of timing specifications).
- Observability still preserved due to disjoint nature of the two switches.

Example 2: Shuttle Abort Sequence Protocol

- Abort QMS/RCS Interconnect (4.18a)
- Presence of both logic and timing dependencies
- 16 steps
- Very sparse protocol (i.e., specifications are based on a large number of inputs and outputs), thus very good observability
- Incomplete specifications
- Generated test sequence and validated under different fault conditions.
- Verified that the sequence in the specification yields a termination of the procedure.

Results

- Validated and verified the following protocols:

X.25

SNA

ABP

LLC 802 Token Ring

- Average saving in the number of tests is almost 20% (compared with Postman)



MISSION
SAFETY CRITICAL
SYSTEMS

RESEARCH & APPLICATIONS

RICIS

SYMPOSIUM '92

27238

P-32

1995109750

202059
578

PERFORMANCE COMPARISON OF TOKEN RING NETWORKS FOR REAL-TIME APPLICATIONS

Sanjay Kamat and Wei Zhao

ABSTRACT

The ability to guarantee the deadlines of synchronous messages while maintaining a good aggregate throughput is an important consideration in the design of distributed real-time systems. In this paper, we study two token ring protocols, the priority driven protocol and the timed token protocol; for their suitability for hard real-time systems. Both these protocols use a token to control the access to the transmission medium. In a priority driven protocol, messages are assigned priorities and protocol ensures that messages are transmitted in the order of their priorities. Timed token protocol does not provide for priority arbitration but ensures that the maximum access delay for a station is bounded.

For both the protocols, we first derive the *schedulability conditions* under which the transmission deadlines of a given set of synchronous messages can be guaranteed. Subsequently we use these schedulability conditions to quantitatively compare the *average case behavior* of these protocols. This comparison demonstrates that each of these protocols has its domain of superior performance and neither dominates the other for the entire range of operating conditions.

BIOGRAPHY

Wei Zhao is currently an Associate Professor at Texas A & M University. He has published extensively in the areas of scheduling algorithms, communications protocols, distributed real-time systems, concurrence control in database systems, and resource management in operating systems. He received the Best Paper Award in the IEEE International Conference on Distributed Computing Systems for a paper on hard real-time communications.

Sanjay Kamat received B.Tech (1985) and M.Tech. (1987) degrees from Indian Institute of Technology, Bombay, India. Currently, he is a Ph.D. candidate in the Department of Computer Science at Texas A & M University. His research interests include distributed systems, real-time systems, and computer networks.

Performance Comparison of Token Ring Protocols for Hard-Real-Time Communication

Sanjay Kamat and Wei Zhao

Department of Computer Science
Texas A&M University
College Station, TX

Presentation Outline

- Introduction
- System Model
- Priority Driven Protocol
- Timed Token Protocol
- Comparison Results
- Conclusions

1. Introduction

- Objective -

To evaluate the performance of two token ring protocols

- The Priority Driven Protocol (eg. IEEE 802.5)

- The Timed Token Protocol (eg. FDDI)

for real-time applications.

Introduction (Cont'd)

Key performance issues in real-time networks

- Guaranty
- Predictability

Introduction (Cont'd)

Three questions to be answered:

1. Given a set of real-time messages and network parameters, will all the messages always meet their deadlines?
 - Schedulability Conditions
2. On average, how high can the load be, before a protocol breaks down?
 - Average Breakdown Utilization
3. Does one protocol perform better than the other?
Under what conditions?

2. System Model

- Network Model
 - * Ring Network with number of nodes = n
 - * BW - Bandwidth
 - * θ - Token walk time (round trip delay)
- Message Model
 - * All real-time messages are synchronous (periodic).
 - * One synchronous message stream per node.
 - C_i - Payload message transmission time
 - C'_i - Augmented message transmission time
(includes overheads)
 - P_i - Message period
 - * Deadline = end of message period
 - * Utilization

$$U = \sum C_i / P_i$$

3. Priority Driven Protocol

- Basic Protocol Description (IEEE 802.5)
 - * Token regulates access to ring.
 - * Messages assigned priorities.
 - * Messages divided into frames.
 - * Token and message frame headers have two priority fields:
 - service priority and reservation priority fields.
 - * Node captures a token if it has messages with priority higher than that of the token.
 - * Other nodes claim the next token via reservation field.
 - * Token holding timer controls maximum number of frames transmitted by a node.
 - * Transmitting node releases a new token with appropriate priority.

Priority Driven Protocol for Real-Time Applications

- The Objective :
To guarantee the deadlines of synchronous messages.
- Problem analogous to processor scheduling for periodic tasks.
- Real-Time scheduling theory.
Rate Monotonic Scheduling (RMS) Algorithm (Liu and Layland)
 - * Optimal static priority algorithm.
 - * Assigns priorities to tasks in inverse relation to periods.
 - * Requires preemption.
 - * Worst Case Achievable Utilization.
(Minimum Breakdown Utilization) is 69%.
 - * Average Breakdown Utilization is approximately 88%.
(Lehoczky, Sha and Ding)

Implementing RMS on a token ring

Proposed by Strosnider and Marchok.

- Assign priorities to synchronous messages according to the Rate Monotonic rule.
- Token Holding Timer at each node set to allow at most one frame transmission.
 - { Leads to priority arbitration at frame level. }
 - { Provides an approximate implementation of preemption }
- Choice of frame size
 - a trade-off between enhanced responsiveness and frame transmission overheads.

Schedulability Criteria for RMS

- Exact schedulability conditions derived by Lehoczky, Sha and Ding for cpu scheduling using RMS.
- Basic idea - for each task, the total demand for resource time should be less than or equal to the available time.
- n Periodic tasks can be scheduled by the rate monotonic algorithm for all task phasings if

$$\forall i, 1 \leq i \leq n, \min_{(k,l) \in R_i} \left(\sum_{j=1}^{i-1} \frac{C_j \lceil lP_k \rceil}{lP_k} + \frac{B_i}{lP_k} \right) \leq 1$$

where $R_i = \{ (k,l) \mid 1 \leq k \leq i, l = 1, \dots, \lfloor P_i/P_k \rfloor \}$
 (Sha, Rajkumar, Lehoczky)

B_i is the worst case **blocking time** of task i .

Extending the schedulability conditions to token ring

We need to do the following

- Compute the augmented message transmission times by accounting for overheads.
 - Overheads associated with frame transmission.
 - Token circulation overheads.
- Compute the worst case blocking time for a message.

Priority Driven Protocol (Cont'd)

- Computing Augmented Message Transmission Times (C_i)
 - * Message divided into frames
Frame transmission time $F = F_{\text{info}} + F_{\text{ovhd}}$
 - * $K_i = \lceil C_i / F_{\text{info}} \rceil$ total number of frames
 $L_i = \lfloor C_i / F_{\text{info}} \rfloor$ number of full length frames
 - * Effective frame transmission time
If $\Theta \geq F$, effective frame transmission time = Θ
{ Since transmitting node has to wait for header to return }

 - Otherwise, it is F
{ at least for all but the last frame }

Priority Driven Protocol (Cont'd)

The final expressions for augmented message transmission time

- Implementation using IEEE 802.5 standard protocol.

$$C'_i = \begin{cases} K_i * \Theta + K_i * \Theta / 2 & \text{when } F \leq \Theta \\ L_i * F + (K_i - L_i) * \max(C_i - L_i * F, \Theta) + K_i * \Theta / 2 & \text{otherwise} \end{cases}$$

Priority Driven Protocol (Contd)

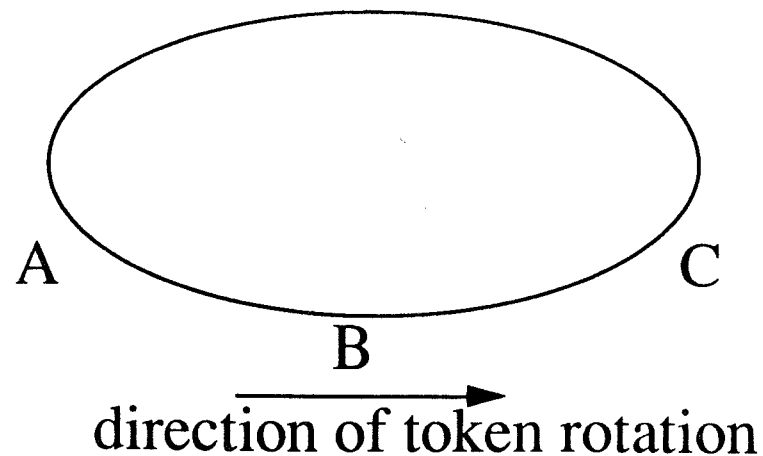
Blocking : Priority inversion due to

- Insufficient priority levels (ignored in this study)
- Approximate nature of preemption
- Bad arrival phasing

Remark:

The worst case blocking interval B_i for any message is $2 * \max(F, \Theta)$.

Example



Schedulability Conditions for Priority Driven Protocol

- Replace task computation times by augmented message transmission times.
- Use worst case message blocking time $B = 2 * \max(F, \Theta)$ in place of B_i .
- n Periodic messages can be scheduled by the priority driven protocol (RMS implementation) for all task phasings if

$$\forall i, 1 \leq i \leq n, \min_{(k,l) \in R_i} \left(\sum_{j=1}^{i-1} \frac{C'_j \lceil lP_k \rceil}{\overline{P}_j} + \frac{B}{\overline{P}_k} \right) \leq 1$$

where $R_i = \{ (k,l) \mid 1 \leq k \leq i, l = 1, \dots, \lfloor P_i/P_k \rfloor \}$

4. Timed Token Protocol

- Amount of time that elapses between channel access times is bounded.
- This bound can be used to calculate the worst case proportion of time available to a node to transmit messages, and hence to guarantee message deadlines.
- Priorityless token is circulated, resulting in a round robin scheduling of transmission.

The Timed Token Protocol (Cont'd .1)

- TTRT (Target Token Rotation Time) gives expected token rotation time.
- $\tau = \Theta + \text{Protocol overheads} \dots$ Denotes the portion of a token rotation that may not be available for synchronous message transmission
- Each node is allocated a portion of TTRT - τ , known as its synchronous capacity (denoted as H_i).
- H_i gives maximum amount of time node i can send synchronous messages each time it receives the token.
- Asynchronous messages can only be sent if actual token rotation time was less than TTRT.

Timed Token Protocol (Cont'd .2)

Guaranteeing messages depends on appropriately allocating the synchronous capacities, H_i .

If H_i is too small, a node may not have enough time to send its synchronous messages.

If H_i is too large, the token rotation time may become too large, e.g., larger than a message period.

Synchronous Capacity Allocation Schemes

- Full length allocation scheme:

$$H_i = C_i$$

- Equal partition, usable portion of TTRT is divided equally among the nodes:

$$H_i = (TTRT - \tau)/n$$

- Proportional allocation scheme:

$$H_i = (TTRT - \tau) * C_i/P_i$$

- Normalized proportional allocation scheme:

$$H_i = ((TTRT - \tau) * C_i/P_i)/U$$

Synchronous Capacity Allocation Schemes (Contd)

- Algorithm to generate optimal H_i 's has been found (Chen et. al)
- Local capacity allocation scheme (Gopal et. al.)

$$H_i = C'_i / (q_i - 1)$$

where

$$q_i = \lfloor P_i / TTRT \rfloor$$

and

$$C'_i = C_i + \lceil C'_i / H_i \rceil * F_{ovhd}$$

All these schemes take TTRT as an input parameter.

Choice of TTRT

Remark : Our studies show that choice of TTRT is critical for obtaining a high real-time utilization without breakdown.

Selecting TTRT as $\sqrt{\tau * P_{\min}}$ is found to give near optimal performance.

The local scheme is found to have a performance nearly as good as the optimal scheme for this choice of TTRT.

Schedulability Conditions for Timed Token Protocol

Any capacity allocation scheme must satisfy two constraints

- Protocol Constraint

$$\sum H_i \leq TTRT - \tau$$

- Deadline Constraint

Minimum time available to transmit a message during its period $X_i \geq C_i$

It has been shown (Chen et al) that

$$X_i = (q_i - 1) * H_i + \max(0, \min(r_i - (\sum_{j \neq i} H_j + \tau), H_i))$$

where $q_i = \lfloor P_i / TTRT \rfloor$ and $r_i = P_i - q_i * TTRT$

Schedulability Conditions for Timed Token Protocol (Cont'd)

Remark: For the local scheme, the deadline constraint is always satisfied.

Hence synchronous messages are guaranteed if and only if the protocol constraint is satisfied.

Hence for the local scheme, schedulability condition is

$$\sum C_i / (q_i - 1) + n * F_{\text{ovhd}} \leq \text{TTRT} - \tau$$

Performance Comparison

Performance Metric

- Minimum Breakdown Utilization

{ Worst case performance }

- Average Breakdown Utilization

{ Average of breakdown loads }

Average Breakdown Utilization presents a better picture of the overall performance of a protocol.

Method for Estimating Average Breakdown Utilization

Sample breakdown loads generated as follows.

- Generate initial message lengths and periods according to specified distribution.
- Uniformly scale up the message lengths till breakdown condition is reached.

5. Comparison

n = number of nodes = 100

d = distance between neighbouring nodes = 100 meters

Average bit delay per station -
4 for priority driven protocol
75 for timed token protocol

Message periods generated using a uniform distribution.
(Maximum to minimum period ratio = 10)

Number of overhead bits per frame = 112

Comparison Results

a) Packet Length = 64 Bytes
Average Period = 10 msec

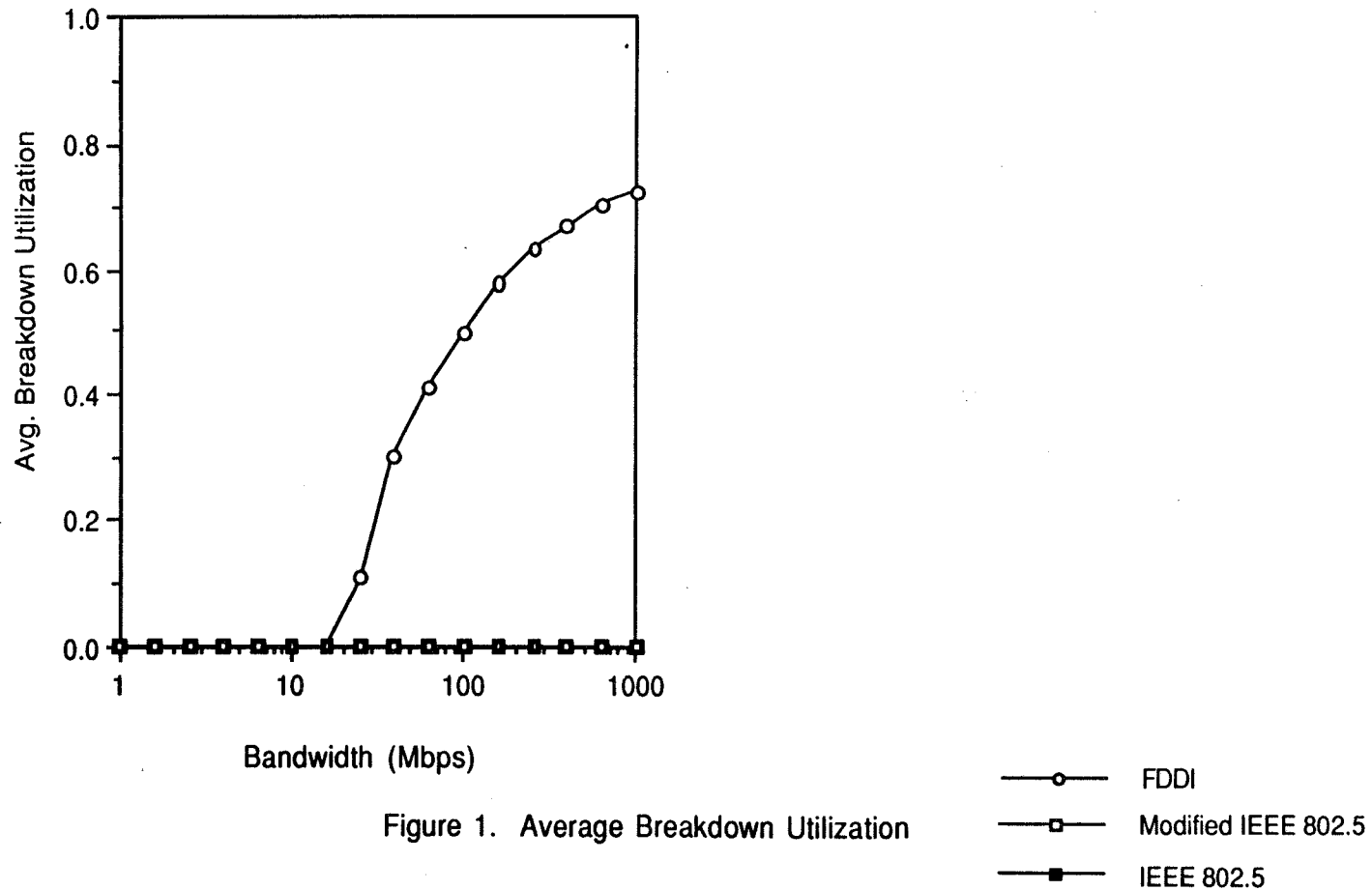


Figure 1. Average Breakdown Utilization

Comparison Results

b) Packet Length = 512 Bytes
Average Period = 10 msec

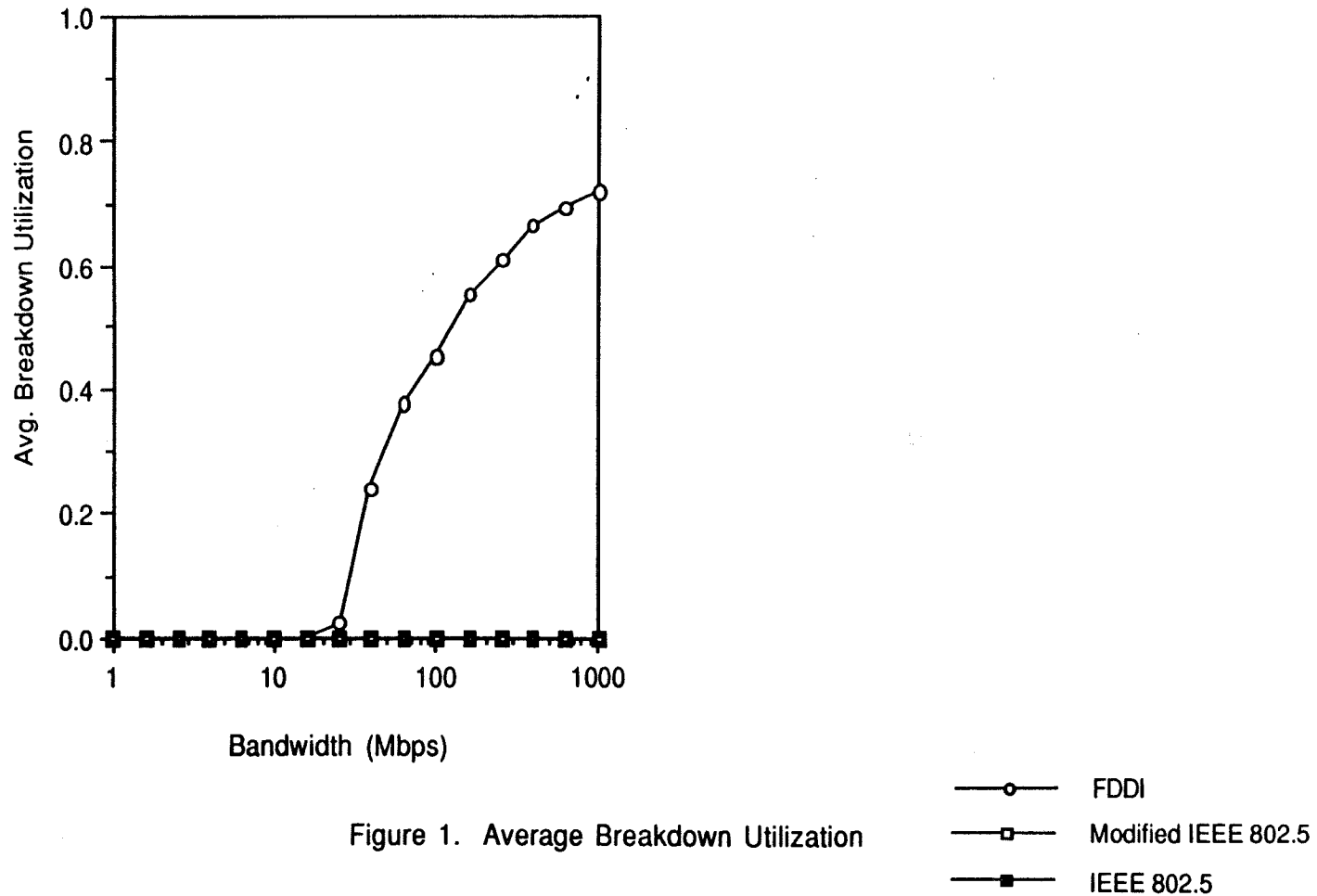


Figure 1. Average Breakdown Utilization

Comparison Results

c) Packet Length = 64 Bytes
Average Period = 100 msec

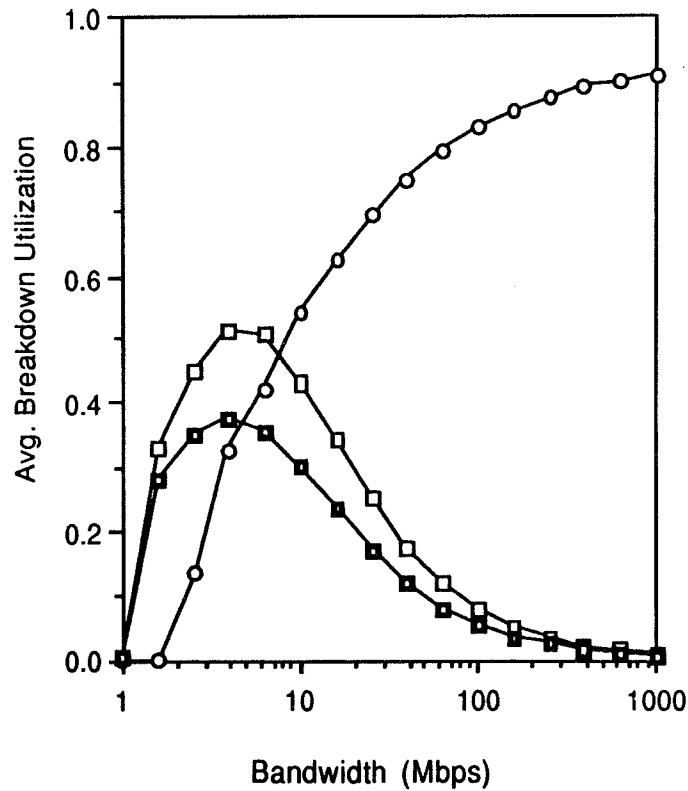
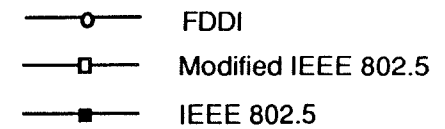


Figure 1. Average Breakdown Utilization



Comparison Results

d) Packet length = 512 Bytes
Average Period = 100 msec

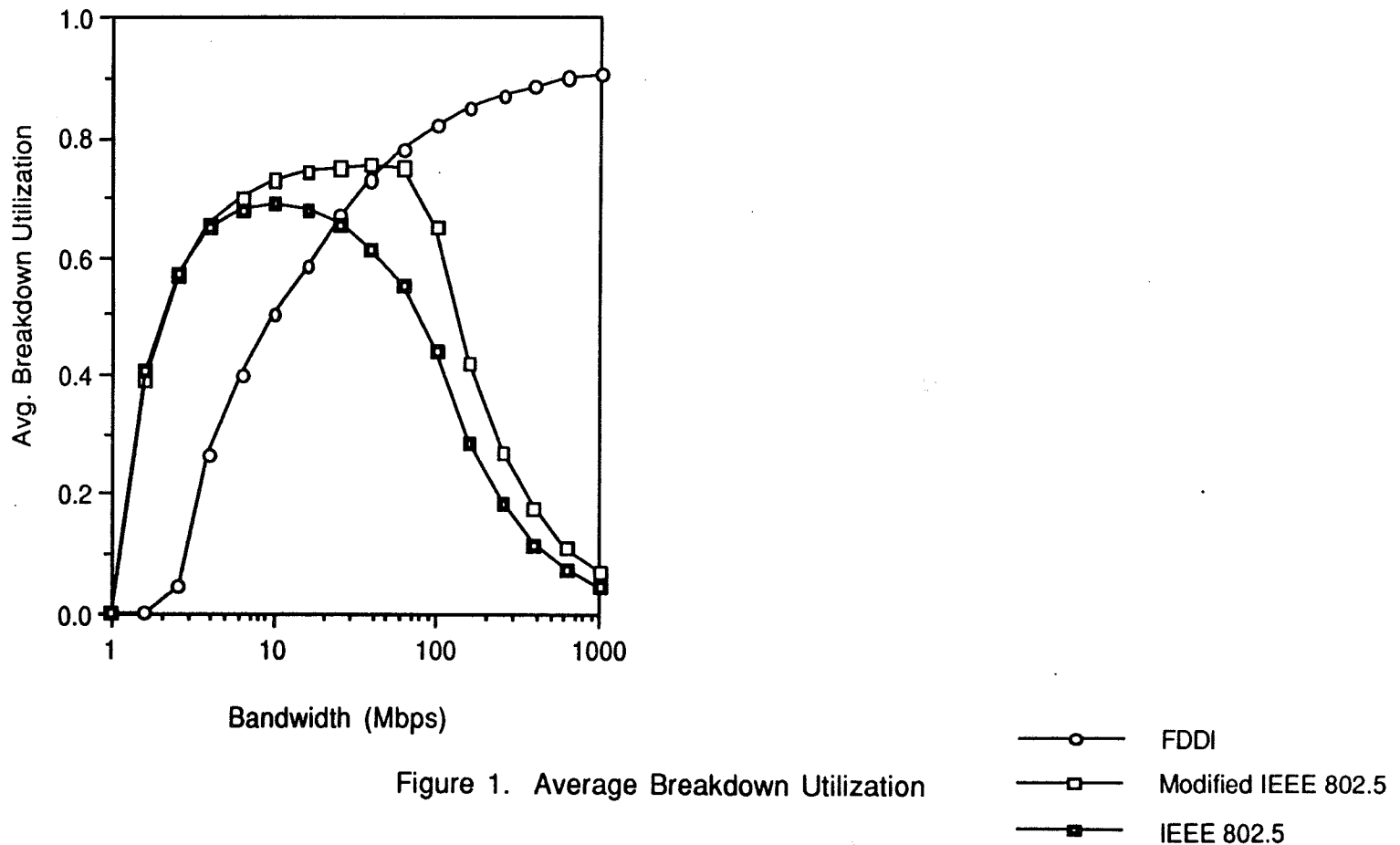


Figure 1. Average Breakdown Utilization

E-D

01

6. Conclusions

- Priority driven protocol and timed token protocol can be adapted for real time applications by suitable choice of protocol parameters
- Schedulability conditions aid operational level network management.
- These conditions can be used to predict the average case performance of protocols.
- Each protocol has its domain of superior performance.

At low transmission speeds the priority driven protocol works better as it efficiently implements optimal scheduling strategy.

At high bandwidths, as the priority arbitration overheads dominate, the timed token protocol works better.

omit

Session IV

SPACE STATION R&D AT RICIS

10:45-12:15

PANEL DISCUSSION

Session Leader:

Lui Sha, *Software Engineering Institute,
Carnegie Mellon University*

Panelists:

Sadegh Davari, *University of Houston-Clear Lake*

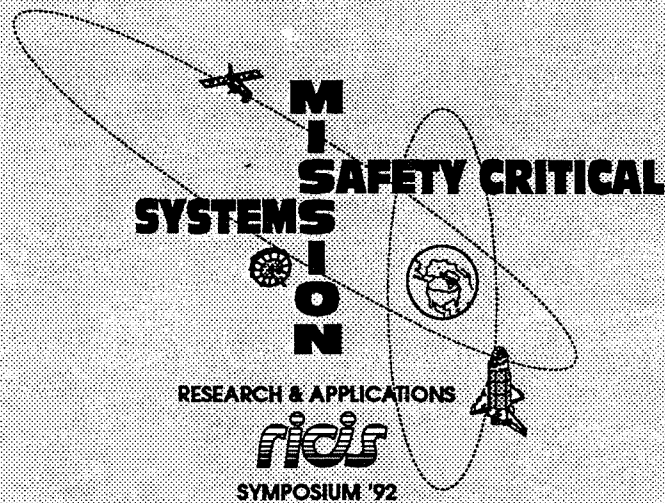
Swaminathan Natarajan, *Texas A & M University*

Wei Zhao, *Texas A & M University*

Frank Miller, *NASA/Johnson Space Center*

John Adams, *IBM Corporation*

Robert Parker, *McDonnell Douglas Space Systems Co.*



PANEL: SPACE STATION R & D AT RICIS

Session Leader: Lui Sha

BIOGRAPHY

Dr. Lui Sha is a senior member of the technical staff at Software Engineering Institute and a member of the research faculty at the School of Computer Science at CMU. He is interested in real-time computing systems. He authored and co-authored a number of articles on the rate monotonic scheduling theory and an analytical approach to real-time software and hardware designs.

Dr. Sha is a member of the IEEE Computer Society. He was the co-chairman of the 1988 IEEE and USENIX workshop on Real-Time Software and Operating Systems. He chairs the Real-Time Task Group of the IEEE Futurebus+ and is the architect of Futurebus+ Real-Time System Configuration guide. He is the Vice Chairman of the 10th International Conference on Distributed Computing Systems in 1990, a consultant to US Navy's Next Generation Computing Resource Program and a reviewer of NASA Space Station Freedom's data management system design.

Dr. Sha received the BSEE degree from McGill University in 1987, the MSEE degree and the Ph.D. degree from Carnegie-Mellon University in 1979 and in 1985.

Sadegh Davari

BIOGRAPHY

Dr. Sadegh Davari is an associate professor of computer science at UHCL. His areas of teaching and research include operating systems, concurrent programming, real-time systems scheduling, and Ada. He is one of the leading researchers in RMS. In conjunction with Dr. Wei Zhao, of TAMU, he received the Best Paper Award in the 12th IEEE International Conference on Distributed Computing Systems held in Yokohama, Japan, June 1992, for a paper on expanding RMS to Distributed Environment.

Swaminathan Natarajan

BIOGRAPHY

Swaminathan Natarajan is an Assistant Professor in the Department of Computer Science at Texas A&M University, College Station, Texas. He received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 1989, his M.S. in Computer Science from the University of Tennessee in 1984, and a B. Tech. in Electrical Engineering from Indian Institute of Technology at Madras, India in 1983. His research interests lie in the field of real-time systems, and programming languages.

Wei Zhao

BIOGRAPHY

Wei Zhao received the Diploma in physics from Shaanxi Normal University, Xian China, in 1977, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, in 1983 and 1986, respectively. He is currently an Associate Professor in the Department of Computer Science, Texas A & M University.

Frank Miller

BIOGRAPHY

Frank Miller is the Subsystem Engineer for space Station Freedom Onboard Local Area Networks with the NASA/Lyndon B. Johnson space Center in Houston, Texas. He received a B.S. in Electrical Engineering and a M.S. in Computer Science from the University of Iowa.

John Adams

Bob Parker

BIOGRAPHY

Bob Parker is Senior Manager for Avionics Verification as a part of the McDonnell Douglas Space Systems Company contract with NASA JSC for the Space Station Freedom Work Package-2 prime contract. He has 27 years experience in aerospace programs performing systems engineering, real-time avionics software development, integration and testing. He has been involved in Space Station avionics multi-system requirements and test since 1988. He obtained a B.S. in Engineering Physics from the University of Illinois in 1961 and a M.S. in EE from the University of California, Irvine in 1975.

Guidelines for Space Station Freedom
Real-Time Software Development & Integration

Presented by
Sadegh Davari, UHCL

Prepared by
Lui Sha & Shirish Sathaye, CMU
Sadegh Davari & Ted Leibfried, UHCL
Wei Zhao & Swami Natarajan, TAMU
Space Station Engineers, MDSSC, IBM, NASA, CSDL

Objectives:

- *Provide simple and practical software architectural guidelines for real-time software development based on the principles of the Rate-Monotonic theory;*
- *Suggest guidelines that are consistent with the majority of the architectural decisions that have been made, while remaining within the framework of the RMS.*

To Achieve the Objectives:

- *Engineers responsible for designing major subsystems were interviewed;*
- *The engineers were asked to suggest topics to be covered in the guidelines;*
- *The engineers were asked to review the draft of the guidelines.*

Topics Covered in the Guidelines:

- *Consideration for a system wide priority assignment scheme,*
- *Tradeoffs in the selection of different task frequencies,*
- *Advantages and potential pitfalls of grouping multiple logical tasks into single software tasks,*
- *Issues of scheduling aperiodic tasks,*
- *Issues in the context of Bus Interface Unit design,*
- *Configuration guidelines for the FDDI network when it is used in real-time applications.*

Recommendations for System-Wide Priorities:

- *Establish a system wide priority assignment table.*

Recommendations for the Selection of Rates:

- *Use only rates that are integer multiple of 0.1 Hz. (hyper period is 10 sec in DMS)*
- *Consider lower rates that are also integer multiples of the basic rates in the system.*
- *Pick the lowest rate subject to the consideration of storage and context switch costs.*

Recommendations for Use of Rate Groups:

- *Tasks of the same program with harmonic frequencies may be combined into rate groups to save context switch and storage costs.*
- *Document the range of rates that are combined together so that potential priority inversion can be detected and dealt with during system integration.*
- *Keep the use of synchronization primitives in source code in the form of comments.*
- *Document considerations such as meeting a particular task's deadline is critical to the system.*

Recommendations for Scheduling Aperiodic Tasks:

- *Process a given class of aperiodic events at the background if the performance is acceptable.*
- *Use polling if background processing cannot deliver the required performance.*
- *Use sporadic server if better performance than polling is needed.*

Recommendations for BIU Software Architecture:

- *Assign priorities according to the generalized rate monotonic algorithm.*
- *If better response time for aperiodic command is needed, consider the use of either polling or sporadic server.*
- *For tasks with very tight jitter requirements, consider the use of harmonic periodics that can be fitted into the highest priority rate group.*

Recommendations for configuring FDDI for Real-Time Applications:

- *To reduce unnecessary changes, the current set ups should be used as long as the FDDI remains schedulable. This is likely since the current bottleneck is at message processing in a node.*
- *If FDDI becomes unschedulable under equal partition, it is advisable to use either synchronous mode to allow for transmitting an allocated number of frames in the asynchronous mode, should the changes are easy to implement.*
- *Consider the use of polling or sporadic server at the application level to guarantee the required performance.*

OVERVIEW

- o RODB CONTAINS VARIOUS DATA ITEMS WHICH ARE ACCESSED AND UPDATED BY APPLICATIONS PROGRAMS
- o BOTH SPACE AND TIME CONSTRAINTS
- o PROPOSED INTERFACE REQUIRED APPLICATIONS PROGRAMMERS TO SUPPLY ADDRESSES FOR DATA TRANSFER
- o CONSTRUCTION OF ADDRESS LISTS ERROR-PRONE, BYPASSES ADA TYPE CHECKING
- o RICIS TEAM DESIGNED AND IMPLEMENTED PREPROCESSOR WHICH PERFORMED TYPE CHECKING, AND AUTOMATED CONSTRUCTION OF ADDRESS LISTS
- o RICIS TEAM ALSO REVIEWED DESIGN OF RODB STORAGE SCHEME AND STUDIED ALTERNATIVES
- o RICIS TEAM ANALYZED RODB ACCESS ALGORITHMS, POINTED OUT POTENTIAL ACCESS CONFLICTS

Complex Object Example *

Set up RODB Interface

```
-- build address list (for POSITIONS parameter in READ_ATTRIBUTE procedure)
```

```
APT_POSITION_ADDRESS_ARRAY : constant STRBD.ADDRESS_ARRAY_T :=
```

```
( APT_POSITION_RECORD.RIGHT_ASCENSION'address,  
  APT_POSITION_RECORD.DECLINATION.DEGREES'address,  
  APT_POSITION_RECORD.DECLINATION.MINUTES'address,  
  APT_POSITION_RECORD.DECLINATION.SECONDS'address );
```

```
-- build attribute list ( for ATTRIBUTE LIST in READ_OPEN procedure) to correspond to HANDLE
```

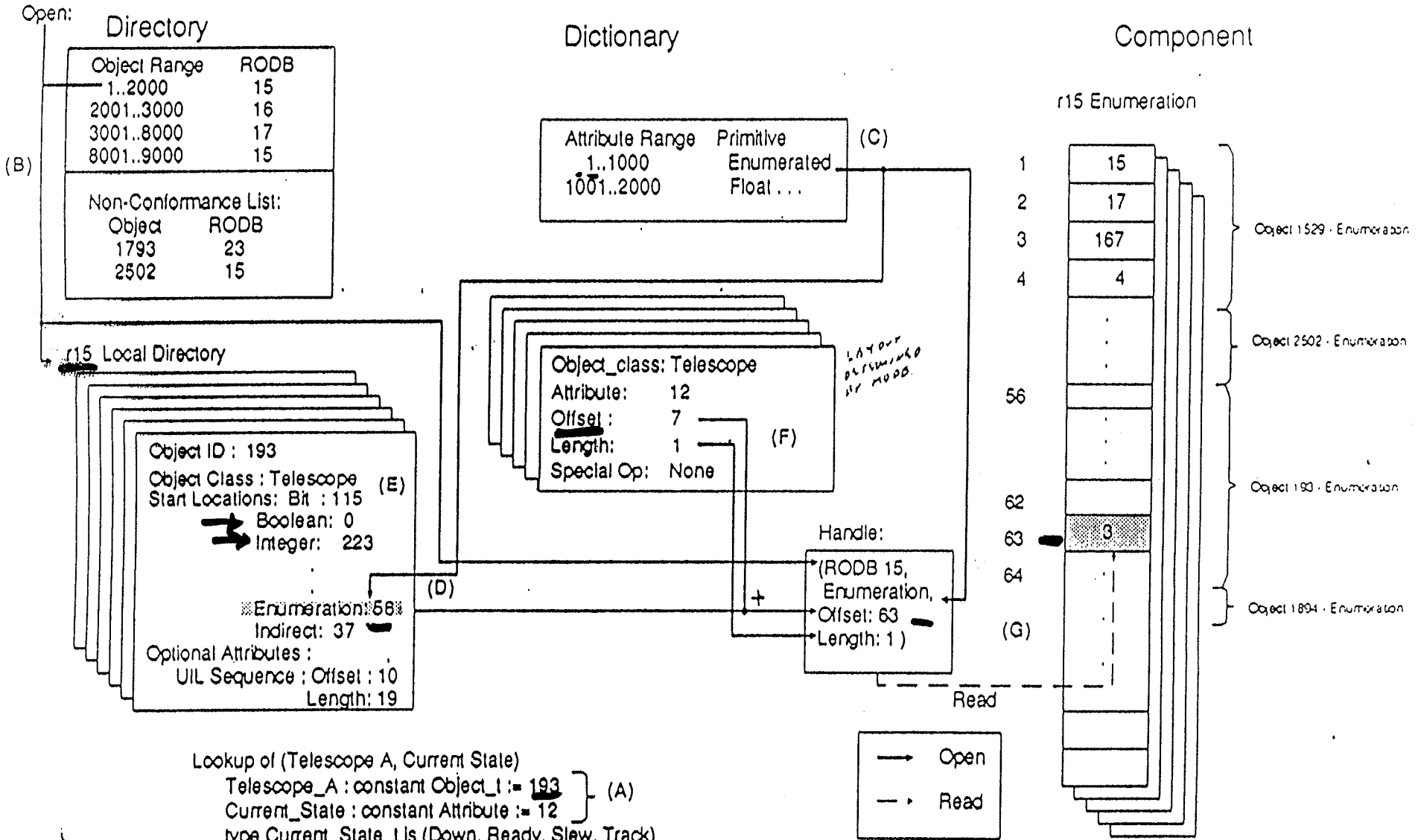
```
APT_POSITION_ATTRIBUTE_ARRAY : constant STRBD.ATTRIBUTE_ARRAY_T :=
```

```
(( AP_TELESCOPE, RIGHT_ASC ), -- each element is a record of CLASS, and ATTRIBUTE  
 ( AP_TELESCOPE, DEC_DEGREES ),  
 ( AP_TELESCOPE, DEC_MINUTES ),  
 ( AP_TELESCOPE, DEC_SECONDS )) ; -- the ATTRIBUTE order is the same as ADDRESSES
```

* IBM CDR example

PREPROCESSOR DESIGN

- o LIST FIELD NAMES WHICH NEED TO BE ACCESSED - EQUIVALENT TO ATTRIBUTE ARRAY (AP_TELESCOPE (RIGHT_ASC, DEC_DEGREES, DEC_MINUTES, DEC_SECONDS))
- o SPECIFY NAMES OF SOURCE/DESTINATION RECORDS (APT_POSITION_RECORD)
- o PREPROCESSOR CONSTRUCTS ADDRESS AND ATTRIBUTE ARRAYS
- o TYPE CHECKING PERFORMED BY CREATION OF AND CALLS TO DUMMY PROCEDURES



RODB STORAGE

- o PROPOSED DESIGN SUGGESTED ALL VALUES OF SAME TYPE STORED TOGETHER, TO AVOID PADDING WASTAGE
- o EXPLORED ALTERNATIVE, INVOLVING STORING ENTIRE RECORD CONTIGUOUSLY
- o CONTIGUOUS STORAGE QUICKER, USES SPACE FOR PADDING
- o DIFFERENCES IN TABLE SPACE NEEDED
- o QUANTITATIVELY ANALYZED TRADEOFF BETWEEN THE TWO SCHEMES

DMS STSV RODB

Flow Control

As described in DMS Det. Des.

← A I P →

Short array of non-specialised tasks OR long array of specialised tasks

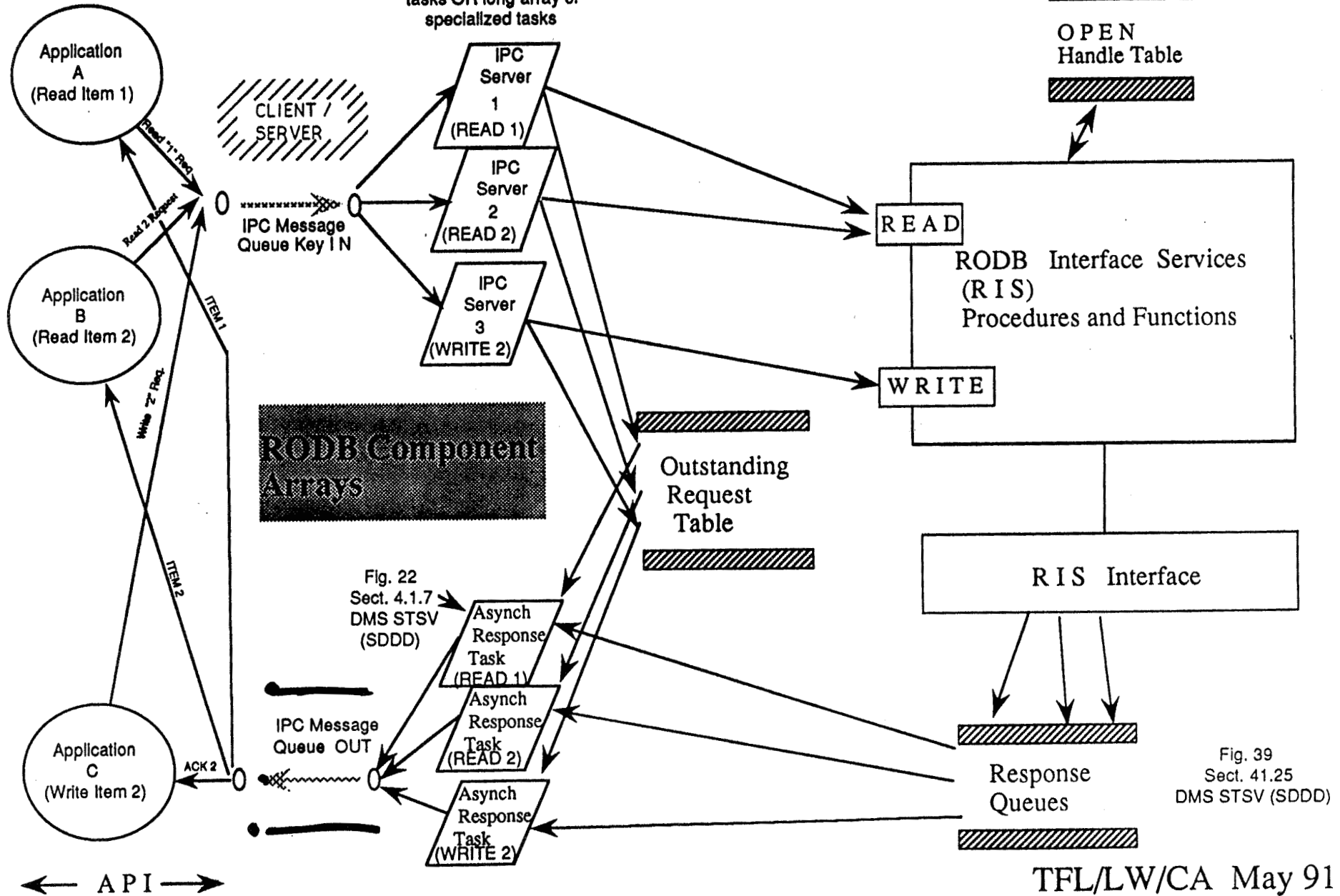


Fig. 39
Sect. 41.25
DMS STSV (SDDD)

RODB ACCESS

- o RICIS TEAM IDENTIFIED POTENTIAL READER/WRITER CONFLICTS IN RODB DATA ACCESS QUEUES
- o THIS PROBLEM WAS RESOLVED IN SUBSEQUENT DESIGN REVISION
- o DETAILED ANALYSIS OF QUEUEING PROCESS ALSO LED TO BETTER UNDERSTANDING OF RODB DESIGN

SSFP AVIONICS TEST AND INTEGRATION CHALLENGES

- **3 Tiers of Asynchronous, Distributed Computers**
 - **Fiber Optic Ring Linked Standard Data Processors (SDP)**
 - **1553 Bus Linked Data Processors (MDM)**
 - **1553 User Bus to Firmware Controllers (FWC)**
- **Multiple Contractors for Development and Integration**
 - **Data Management System - Work Package 2 (WP-2)**
 - **Electrical Power - WP-4**
 - **External Thermal Control - WP-2**
 - **Internal Thermal & Environment Control - WP-1**
 - **Other Core Avionics (Guidance, Communications, Etc) - WP-2**
 - **Laboratory Payload Accommodation - WP-1**
- **Mix of Software**
 - **Commercial, Off-The-Shelf, Software (COTS)**
 - **Adaptations of COTS**
 - **New SSFP Unique Software**

SSFP AVIONICS TEST AND INTEGRATION KEY OBJECTIVES

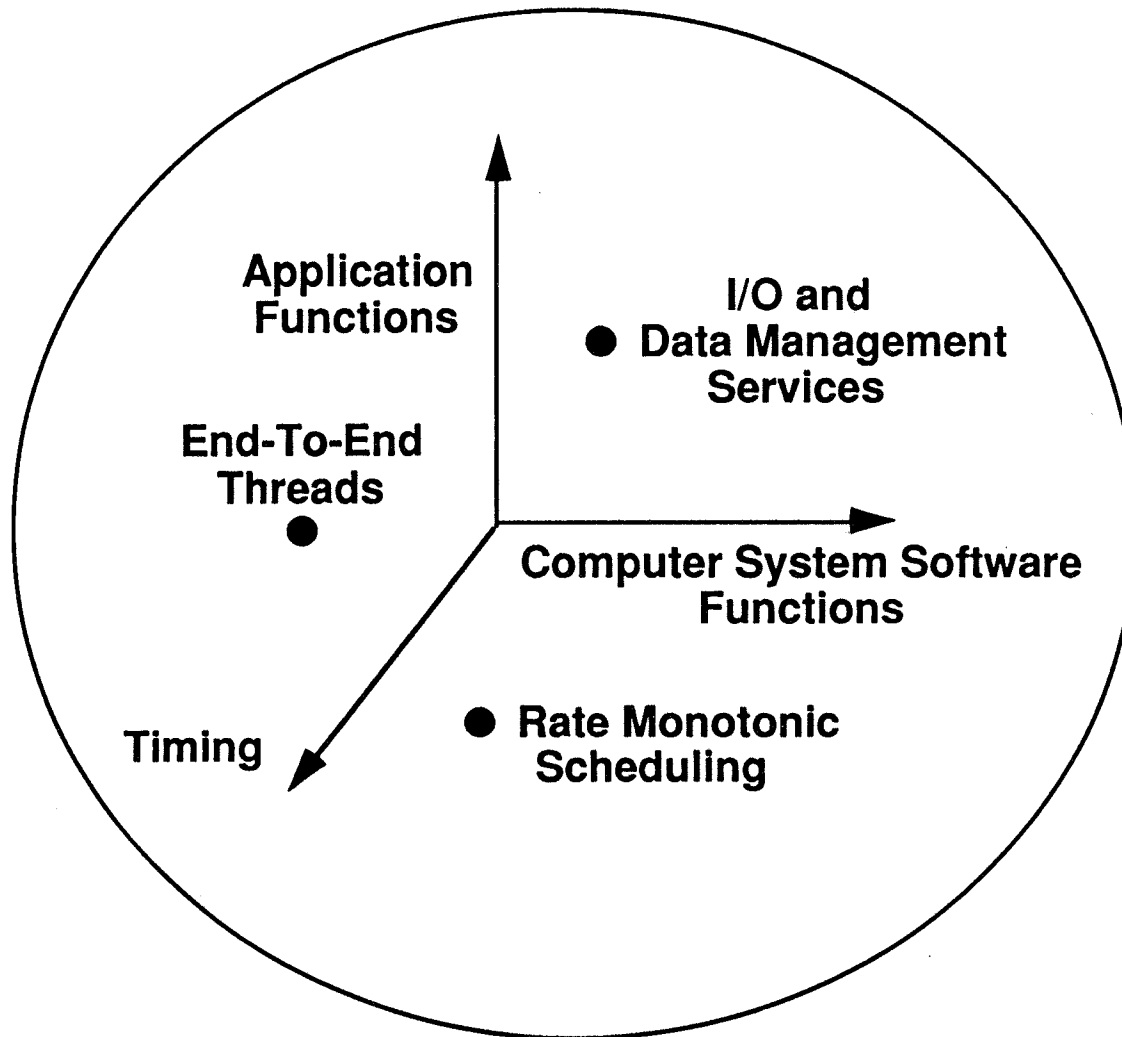
■ Effectiveness

- Early Detection of All Critical Problems
- Rapid, Accurate Isolation of Discrepancies
- Progressive Reduction of Discrepancy Density
- Convergence of Test Success with Integration and Launch Schedule

■ Productivity

- High Throughput in Test Facility
 - Batch Processing of Tests
 - Minimize Test Set-up Time
 - Efficient Off-Line Support (Post Processing, Etc.)
- Minimize Manpower for Repetitive, Standard Tasks
- Minimize Retest Needs, Efficient Regression Testing

SSFP TEST AND INTEGRATION TECHNOLOGY



omit

Session V

LANGUAGE & SYSTEM STANDARDS

1:30 - 3:00

Session Leader:

Swaminathan Natarajan, *Texas A & M University*

Presenters:

POSIX Real-Time Extensions

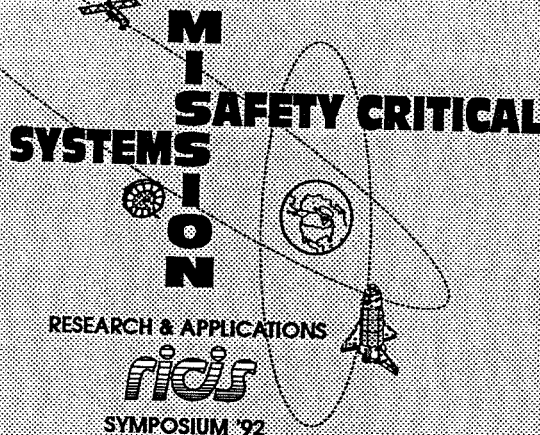
Henry H. Robbins, *IBM Federal Systems Company*

Ada_9X Overview

David G. Weller, *CAE Link Corp.*

CIFO 3.0

Pat Rogers, *SBS Engineering*



59-6

27239

p-31

1995 107751

324060

54p.

POSIX REAL-TIME EXTENSIONS

Henry H. Robbins

ABSTRACT

POSIX is an evolving set of operating system interface standards, whose parts are in varying stages of production in a number of standards working groups. This presentation separates the real-time POSIX standards work in progress from the rest and provides an overview of the purpose, status, dependencies, content, and projected schedule of each.

BIOGRAPHY

Mr. Robbins is an Advisory Systems Engineer with the IBM Federal Systems Company. His recent work includes software engineering support of NASA JSC's Space Station Control Center, which requires the use of POSIX operating systems and the Ada language. He is also IBM's representative to the IEEE TCOS 1003.5 (Ada Bindings) Working Group, which is currently addressing Ada bindings to real-time POSIX.

POSIX Real-Time Extensions

October 29, 1992

Henry H. Robbins

IBM Federal Systems Company
3700 Bay Area Blvd.
Houston, Texas 77058

(713) 335-3297

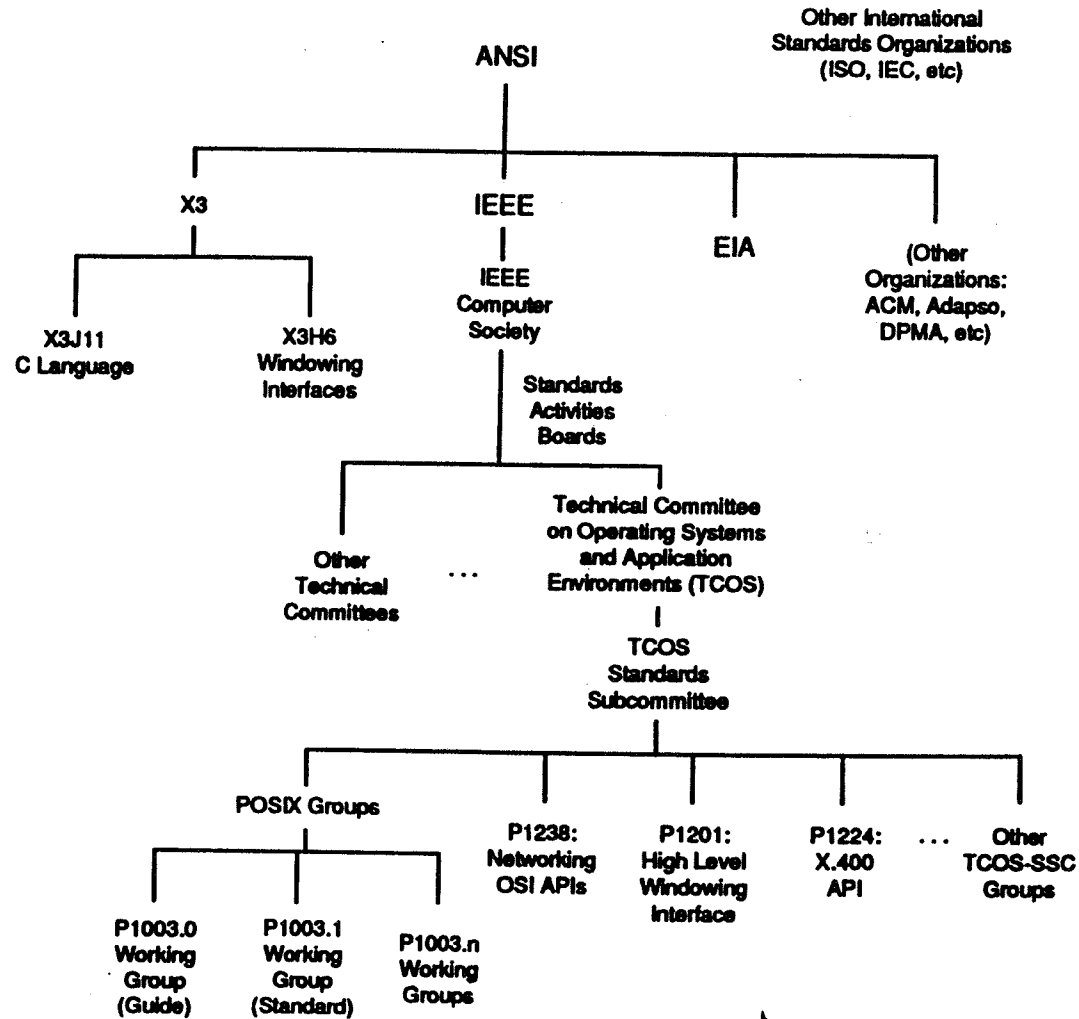
robbinsr@houvmssc.vnet.ibm.com

The IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font with horizontal stripes through the letters.

1. What is POSIX?

- POSIX stands for **P**ortable **O**perating **S**ystem **I**nterface for **(X)** Computer Environments
- POSIX defines a standard operating system *interface* and environment to support application portability at the source code level.
 - The interface is UNIX-like, but does not require an underlying UNIX operating system
 - The POSIX Open Systems Environment has the goals of:
 - Source Code **Portability** through system/application interface standards
 - **Interoperability** of applications resident on dissimilar computer systems
 - **User Portability** from system to system and between applications on the same system with a minimum of retraining
- In the U.S., POSIX standards are written by working groups of the IEEE Computer Society's Technical Committee on Operating Systems

2. IEEE Standards Diagram



3. IEEE TCOS POSIX Standards Process

- Working Group submits a Program Authorization Request (PAR) to IEEE TCOS
- Draft standard written
- Mock Ballot on draft standard
- Modified draft standard
- Balloting group formed
- Formal ballot on draft standard
 - 75% approval required (plus other rules)
 - Otherwise iterate
- Submit to IEEE Standards Board for Approval

4. POSIX Standards and Drafts

Standard	Description	Status
P1003.0	Guide to Open Systems Environment	1st Ballot Complete
P1003.1	System API (C Binding)	Approved (POSIX .1-1990)
P1003.1-LI	Language Independent System API	Currently in 1st Ballot
P1003.1a	System API Extensions	1st Ballot - 1993
P1003.2	Shells and Utilities	Approved - Sept. 1992
P1003.2b	Shells and Utilities - Amendment 2	Initial Work
P1003.3	Test Methods	Approved (POSIX .3-1991)
P1003.3.1	Test Methods for 1003.1	Final Ballot Complete
P1003.3.2	Test Methods for 1003.2	1st Ballot - 4Q92
P1003.4	Real Time and Related System API	Final (?) Ballot - 4Q92
P1003.4a	Threads API	Next Ballot - 1/93
P1003.4b	System API Extensions	Next Ballot - 1/93
P1003.4c	Language Independent Spec	Initial Work - 1/93
P1003.5	Ada Bindings (System API)	Approved - June 1992
P1003.6	Security	Next Ballot - 4Q92
P1003.7.1	Print Administration	1st Ballot - 1/93
P1003.7.2	Software Administration	Mock Ballot - 1Q93
P1003.7.3	User Administration	Initial Work (no PAR)
P1003.8	Transparent File Access	Next Ballot - 10/92

5. POSIX Standards and Drafts (Continued)

Standard	Description	Status
P1003.9	FORTRAN 77 Bindings	Approved - June 1992
P1003.10	Supercomputing Appl. Environment Profile	1st Ballot - 10/92
P1003.11	Transaction Proc. Appl. Environment Profile	In Ballot Resolution
P1003.12	Protocol Independent Interface	Mock Ballot 4Q92
P1003.13	Real Time Appl. Environment Profile	1st Ballot - 10/92
P1003.14	Multiprocessing Appl. Environment Profile	1st Ballot - 4Q92
P1003.15	Batch Environment Amendments	1st Ballot - 4Q92
P1003.16	C Language Bindings for 1003.1-LI	In Ballot
P1003.17	Directory Server API	In Ballot
P1003.18	Platform Environment Profile	1st Ballot soon
P1003.19	FORTRAN 90 Binding to 1003.1-LIS	Initial Work
P1003.20	Ada Bindings (Real Time)	Mock Ballot 1/93
P1003.20a (?)	Ada Bindings (Threads API)	Initial Work (PAR in progress)
P1201.1	Window Intf. for User and Appl. Portability	1st Ballot - 1993
P1201.2	General User Interfaces - Drivability	In Ballot Resolution
P1224	ASN.1 Object Management API	In Ballot Resolution
P1224.1	X.400 API	In Ballot Resolution
P1238.0	Connection Management API	1st Ballot - 2Q93
P1238.1	FTAM API	Initial Work

6. Real Time POSIX PARs

Draft Standard	Description
1003.4 Working Group	
P1003.4	Real Time and Related System API
P1003.4a	Threads API
P1003.4b	System API Extensions
P1003.4c	Language Independent Spec
P1003.13	Real Time Appl. Environment Profile
1003.5 Working Group	
P1003.20	Ada Bindings (Real Time)
P1003.20a (?)	Ada Bindings (Threads API)

7. Terminology - Application Environment Profile

- Most POSIX standards are in one of two flavors:

Application Programming Interface (API) The interface between the application software and the application platform, across which all services are provided. P1003.4, P1003.4a, and P1003.4b are all APIs. APIs specify interface standards, but operating systems do not have to implement all of their specified services.

Application Environment Profile (AEP) A profile, specifying a completed and coherent specification of the POSIX Open Systems Environment, in which the standards, options and parameters chosen are necessary to support a class of applications. P1003.13 is an AEP.

- The intent is for operating systems to claim compliance to an AEP, instead of APIs.

8. Terminology - Language Independent Service Specification (LIS)

- An API specification that facilitates the management and development of consistent language binding standards
- Originally, all POSIX API specifications were to be in the C language, with other languages, like Ada, binding to C.
- Several years ago, the rules were changed to require language-independent API specifications with a separate C-language binding to it. Other languages, like Ada, would bind to the LIS.
- P1003.1 (System API (C Binding)), P1003.4 (Real Time and Related System API), and P1003.5 (Ada Bindings (System API)) were "grandfathered" and follow the old rules.
- **Note: P1003.20 (Ada Bindings (Real Time) - not "grandfathered") cannot bind to P1003.4, which is a "grandfathered" C API specification.**



9. P1003.4 Real Time and Related System API

- P1003.4 is the *process-level* API specification for Realtime services. It does not support concurrency within a POSIX process.
- It enhances and adds to the services specified by P1003.1.
- Current Draft is Draft 12.
- Ballots on draft 12 presently stand at 70% approval with many abstentions for lack of time. Most of the objections were on *seminit()*.
- The chairman of the working group expects to be able to get over 75% approval during ballot resolution.
- The working group currently expects to have a circulation of 10-15 pages of updates (only) for ballot in October 1992.
- The December 1992 date for adoption of .4 is optimistic. A more likely date would be March 1993.

10. P1003.4 Real Time and Related System API (Contents)

Revisions to Process Primitives: Contains the realtime signal enhancement to assure signal delivery by queuing signals. P1003.1 signals are not queued and may be lost if another signal follows.

Revisions to Process Environment: Adds to the list of P1003.1 configurable system variables.

Revisions to Files and Directories: Augments I/O primitives defined in P1003.1 and P1003.1a.

Revisions to Input and Output Primitives: Augments P1003.1 Synchronous I/O to guarantee I/O completion to the physical medium. Also supplies Asynchronous I/O, permitting the calling process to continue as soon as the I/O is queued; completion notification is via a POSIX signal.

11. P1003.4 Real Time and Related System API (Contents Continued)

- Synchronization:** Provides *counting* semaphores. This changed from binary semaphores in Draft 12. This area still needs some work.
- Memory Locking:** Standardizes modern UNIX practice providing the ability to lock physical memory from paging and swapping.
- Memory Mapped Files and Shared Memory:** A shared memory facility similar to the *mmap* interface in AT&T's System V, plus an interface for generalized file mapping.
- Execution Scheduling:** Provides FIFO and round-robin priority scheduling, as well as an interface to set the scheduling method.

12. P1003.4 Real Time and Related System API (Contents Continued)

- Clocks and Timers:** A time service loosely based on Berkeley interval timers, allowing for completely time-driven processing.
- Interprocess Message Passing:** Provides priority Message Queues with signal notification.
- Realtime Files:** Provides a mechanism to manipulate realtime attributes of files, such as the preallocation of disk blocks and direct unbuffered I/O to a disk file.

13. P1003.4 Real Time and Related System API (Schedule)

Milestone	Projected Date
Expect Next Draft	October 1992
Next Ballot	October 1992
Standard Approved	December 1992

14. P1003.4a Threads API

- P1003.4a is the *thread*-level API specification for Realtime services. It does support concurrency within a POSIX process.
- It enhances and adds to the services specified by P1003.1 and P1003.4.
- It was split off from P1003.4 because of the difficulty in reaching consensus due to varying goals within the working group, especially between Realtime and Multiprocessor advocates.
- Current Draft is Draft 6.
- Ballot objections against draft 6 are at 5000, up from 3000 against the previous draft.
- These objections are being resolved and the working group expects the next draft around the end of 1992.
- P1003.4a must still be converted to a LIS.

15. P1003.4a Threads API (Contents)

- Definitions and General Requirements:** Extensions to definitions, error numbers, data types, numerical limits, and symbolic constants to support threads
- Thread Management:** Creation and termination of threads, plus thread ID operations
- Synchronization Primitives:** Operations on Mutexes and condition variables
- Thread-Specific Data:** Key/value mechanism to associate an opaque key with each per-thread datum. These keys play the role of identifiers for per-thread data.
- Thread Priority Scheduling:** Operations on the attributes of thread scheduling. This includes the setting of priority ceilings for Mutexes.

16. P1003.4a Threads API (Contents Continued)

- Process Control:** Operations to support *fork()* and *exec()* on processes in a threads environment.
- Signals:** Extensions to realtime signals in a threads environment, including per-thread signal masks, per-process signal vectors, and single delivery of each signal
- Thread Cancellation:** Operations that allow the cancellation of another thread
- Reentrant Functions:** Extensions to other P1003.1 and P1003.4 routines to allow them to work in a multi-threaded environment



17. P1003.4a Threads API (Schedule)

Milestone	Projected Date
Expect Next Draft	December 1992
Next Ballot	January 1993
Standard Approved	December 1993

18. P1003.4b System API Extensions

- P1003.4b provides additional realtime services
- These services are considered to be at a lower priority than those in P1003.4 and P1003.4a.
- Current Draft is Draft 4.
- Recent developments:
 - The proposed chapter on asynchronous services was dropped because nobody was interested in the function.
 - The working group created one new chapter (Device Control), made changes to all four existing chapters, and created corresponding LIS chapters.
- Standard approval is several years away

19. P1003.4b System API Extensions (Contents)

- Process Primitives:** *spawn()*: *fork()* followed by *exec()* in one syscall
- Timeouts for blocking services:** Additional interfaces for blocking real time services are provided. These interfaces allow the specification of a time out value. The service is not allowed to block longer than the specified amount of time.
- Execution Time Monitoring:** Accumulate CPU time used by individual threads and processes
- Sporadic server:** Introduces a scheduling class for aperiodic events
- Device control:** Defines a standard way for controlling non-standard devices, similar to *ioctl()* but less controversial

20. P1003.4b System API Extensions (Schedule)

Milestone	Projected Date
Expect Next Draft	September 1992 (A)
Next Ballot	January 1993
Standard Approved	?



21. P1003.4c Language Independent Spec

- P1003.4c will be the Language Independent Services Specification corresponding to P1003.4 (Real Time and Related System API).
- The 1003.5 (Ada Bindings) working group is the group primarily interested in the LIS for .4 since, by the requirements of the P1003.20 PAR, it cannot bind to the existing P1003.4.
- In the July meeting, Offer Pazy became the book manager.
- A P1003.4c draft is needed by next spring to meet the P1003.20 schedule.

22. P1003.13 Real Time Appl. Environment Profile

- P1003.13 consists of four profiles ranging from a minimal system for unattended control to a large multi-purpose system.
- Current Draft is Draft 5.
- The first ballot (draft 5) produced over 75% approval. BUT:
 - The group has to wait until .4 and .4a, the underlying standards, have been approved by the IEEE.
 - The group has to split the profiles into four separate documents, one for each profile.
- There was an interesting flap in the July meeting over a proposed TCOS requirement that all application environment profiles include all of P1003.1 and P1003.2. Most realtime folks fail to see the utility of a user shell in the head of a missile.



23. P1003.13 Real Time Appl. Environment Profile (Contents)

Minimal Realtime System Profile: Embedded system dedicated to unattended control of one or more special I/O devices. No user interaction or file system required. Supports a single P1003.1 process with one or more POSIX.4a threads and perhaps an IPC interface for communication with like systems.

Realtime Controller System Profile: Like minimal-profile systems, with the addition of a file system interface, character by character serial I/O interfaces, asynchronous (non-blocking) I/O interfaces, and P1003.1 signals

Dedicated Realtime System Profile: An extension of the minimal-profile system, with support for multiple processes and most P1003.4 functionality except Realtime Files. There is a common interface for device drivers and files, but no hierarchical file system.

24. P1003.13 Real Time Appl. Environment Profile (Contents Continued)

Multi-Purpose Realtime System Profile: Provides comprehensive functionality and runs a mix of differing realtime and non-realtime tasks. Includes P1003.1, P1003.2 (user shell), and P1003.4. P1003.4a threads support is optional. Additional functionality is provided by options for networking, windowing, and programming languages.



25. P1003.13 Real Time Appl. Environment Profile (Schedule)

Milestone	Projected Date
Expect Next Draft	October 1992
Next Ballot	October 1992
Standard Approved	?

26. P1003.20 Ada Bindings (Real Time)

- Currently a thin, direct binding to P1003.4, based on Dr. Ted Baker's work for the Army. Contents are the same as for P1003.4.
- A mock ballot is scheduled for January.
- Before P1003.20 can enter the formal balloting process, it will have to become a binding to P1003.4C, the Realtime Language Independent Services Specification.
- Draft 0.5 of P1003.20 held up very well in the July meeting under very close scrutiny, and very few changes were recommended. The group discussed many potential issues, but only recommended a few minor changes in the areas of clocks and timers, plus three issues to present to the 1003.4 working group in clocks and timers and in message queues.
- P1003.20 will be a binding to Ada 83, perhaps with some extensions to take advantage of Ada 9x capabilities.



27. P1003.20 Ada Bindings (Real Time) Schedule

Milestone	Projected Date
Mock ballot mailing (binding to P1003.4)	Oct. 1992
Mock ballot	Jan. 1993 - Feb. 1993
Mock ballot resolution	Mar. 1993 - July 1993
1st ballot (binding to .4 LIS)	Oct. 1993 - Dec. 1993
1st ballot resolution	Jan. 1994 - July 1994
1st recirculation ballot	Aug. 1994 - Oct. 1994
1st recirculation ballot resolution	Jan. 1995 - April 1995
2nd recirculation ballot	May 1995 - June 1995
Approved draft standard to IEEE	Sept. 1995

28. P1003.20a (?) Ada Bindings (Threads API)

- P1003.20a will be an Ada binding to the LIS equivalent of P1003.4a (Threads API). Schedule tbd; name may change when PAR is approved.
- The July decision to move the work of binding to P1003.4a to a separate PAR was made to accommodate the projected schedule for P1003.4a and the resulting LIS, which is estimated to be at least two years behind the P1003.4 schedule. The 1003.5 group intends to split its time between P1003.20 and P1003.20a.
- The working group will concentrate on a threads model in which all threads created by Ada are created by the RTE as Ada tasks. This eliminates the need to bind to *pthread_create()*, *pthread_join()*, *pthread_detach()*, *pthread_cancel()*, and *pthread_exit()*.
- The working group also decided that the problems of Ada thread inter-operation with C threads were too complex for the standard: for example, if a bound C subroutine creates a thread, can Ada rendezvous with it? Any such facilities will be left up to the implementers.

29. Suggested Reading

- In the October 1992 issue of Embedded Systems Programming (pp 28-40):
 - “Real-Time POSIX”, by Bill O. Gallmeister (vice-chair of the 1003.4 working group)
- For POSIX.1 basics:
 - Lewine, Donald, *POSIX Programmers Guide*, Sebastapol, Calif.: O’Reilly and Associates, 1991

Ada 9X Overview
ABSTRACT

27240
307064 P-7 79.
1995 107752

The current version of Ada has been an ANSI standard since 1983. In 1988, the Ada Joint Program Office was tasked with reevaluating the language and proposing changes to the standard. Since that time, the world has seen a tremendous explosion in object-oriented languages, as well as other growing fields such as distributed computing and support for very large software systems. Mr. Weller will discuss the new features being added to the next version of Ada, currently called Ada 9X, and what transition issues must be considered for current Ada projects. The presentation assumes a familiarity with the features of the current Ada programming language.

BIOGRAPHY

Mr. Weller is a senior systems engineer with CAE-Link, Space Technology Division. He is the project leader of the Software Engineering Group, which is responsible for the definition of the software architecture and development methodology for both the Space Station and Space Shuttle Training Systems. Mr. Weller has been working with Ada since 1985, and is currently an official reviewer of the Ada 9X language. Mr. Weller was previously in the Air Force in the Electronic Warfare arena.

The New Face of Ada



- Programming Paradigms
- Multitasking and Parallel Processing
- Distributed Processing
- Programming-in-the-Large
- Specialized Needs
- Object-Oriented Programming
- Ada 9X compared to C++ 3.0
- Transition Issues

Programming Paradigms

- International Support
- Subprogram Parameters
- "Foreign Language" Support
- Storage Allocation/Reclamation
- Generics
- Exception Handling
- I/O Support

Multitasking and Parallel Processing

- Task Creation and Destruction
- Protected Records
- Massively Parallel Architectures
- Vector Processing

Distributed Processing

- Partitions
- Dynamic Reconfiguration
- User Defined Communication Package (UDCP)

Programming-in-the-Large

- Avoiding Recompilation
- Subsystems
- Incremental Development

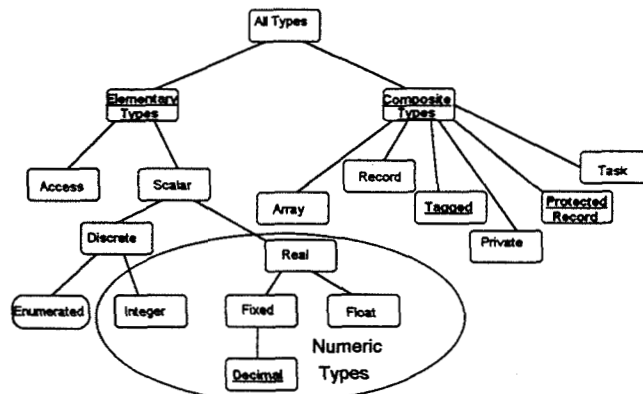
Specialized Needs

- Systems Programming
- Safety-Critical and Trusted Applications
- Information Systems
- Scientific and Mathematical Systems

Object-Oriented Programming

- Type Hierarchy
- Type Classes/Inheritance
- Operations and Overloading
- Polymorphism
- Multiple Inheritance

Type Hierarchy



Ada 9X compared to C++ 3.0

		Ada 9X	C++ 3.0
<i>Abstraction</i>	Instance variables	Yes	Yes
	Instance Methods	Yes	Yes
	Class variables	Yes	Yes
	Class Methods	Yes	Yes
<i>Encapsulation</i>	Of variables	Public, protected, private	Public, protected, private
	Of methods	Public, protected, private	Public, protected, private
<i>Modularity</i>	Kind of Modules	Package	File (header/body)
<i>Hierarchy</i>	Inheritance	Yes, partial multiple	Multiple
	Generic units	Yes	No
	Metaclasses	Yes	Yes (templates)
<i>Typing</i>	Strongly typed	Yes	Yes
	Polymorphism	Yes (single)	Yes (single)
<i>Concurrency</i>	Multitasking	Yes (synch or asynch)	Yes (defined by class)
<i>Persistence</i>	Persistent Objects	No (Streams supported)	No (Streams supported)

Transition Issues

- New Reserved Words
- Implicit Assumptions
- Static Literals
- Ada 9X Publications
- Validation rules for 9X
- Compiler Availability

Where Can I Learn More?

- Anonymous ftp from `ajpo.sei.cmu.edu` (go to `/pub/ada9x` directory)
- Ada 9X BBS: 1-800-Ada-9X25
- Ada Information Clearinghouse
IIT Research Institute
4600 Forbes Blvd
Lanham, MD 20706-4312
- Ada 9X Project Office
PL/VTET
Kirtland AFB, NM 87117-6008



27241
p-15
1995107753
324066
16p.

CIFO 3.0

Pat Rogers

ABSTRACT

The Ada Runtime Environment Working Group (ARTEWG) has, since 1985, developed and published the Catalog of Interface Features and Options (CIFO) for Ada runtime environments. These interfaces, expressed in legal Ada, provide "hooks" into the runtime system to export both functionality and enhanced performance beyond that of "vanilla" Ada implementations. Such enhancements include high- and low-level scheduling control, asynchronous communications facilities, predictable storage management facilities, and fast interrupt response. CIFO 3.0 represents the latest release, which incorporates the efforts of the European realtime community as well as new interfaces and expansions of previous catalog entries. This presentation will give both an overview of the Catalog's contents and an "insider's" view of the Catalog as a whole.

BIOGRAPHY

Pat Rogers is a Consulting Scientist at SBS Engineering in Houston, where he is the principal investigator for a project which has developed a Distributed Ada implementation for the U.S. Air Force. He has been involved with Ada since 1980, is a founding member of ARTEWG, and is a contributing member of the CIFO development subgroup.

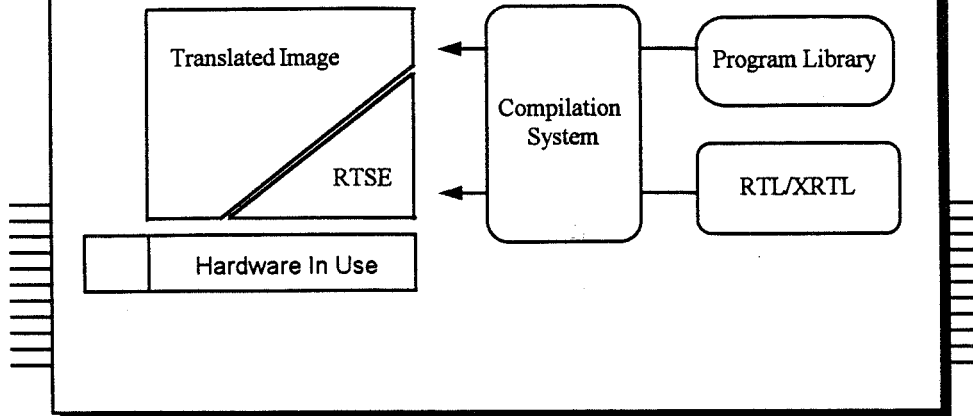
The New CIFO : Bridging Ada83, Realtime Systems and Ada9X

Pat Rogers
SBS Engineering, Inc.
18333 Egret Bay Blvd., Suite 340
Houston, Tx 77058
713/333-5040

What Is CIFO?

- Catalog of Interface Features and Options
- A set of common packages, subprograms and pragmas used to extend the capabilities of the Ada baseline facilities (the RM) via the runtime environment
- Developed by the Ada Runtime Environment Working Group -- ARTEWG ("art-wig")
- Began in 1985, first working meeting at UHCL
 - Dr. Charles McKay is chair of team
- Many Ada compiler vendors supporting CIFO
- Many users, including Space Station Freedom contractors

The Extended Runtime Library



Structural Differences From 2.1

- Reorganized Entry groupings
- New sections per Entry
 - Interactions With Other Entries
 - Changes From The Previous Release
- Interactions Matrix

Convergence with ExTRA

- Extensions des services Temps Reel Ada
 - *Ada Real Time Service Extensions*
- Many Entries added
- A few Entries superseded
- Finalized the change in CIFO orientation

Change In Orientation

- Early CIFO releases
 - Targeted to hard-constrained applications
 - Open to abuses and misuses
- Later CIFO releases
 - Less specific to hard-constrained applications
 - Abuses covered, but at what price?

Functional Categories

- Basic Mechanisms
- Scheduling Controls
- Asynchronous Cooperation Mechanisms
- Interrupt Support
- Compiler Directives
- Memory Management Mechanisms

☞ Indicates a new entry in following pages

Basic Mechanisms

- Task Identifiers
- ☞ Queuing Discipline
 - Provides basic definitions for task entry and scheduling disciplines

Scheduling Control

- Synchronous & Asynchronous Scheduling
- ☞ Priority Inheritance Discipline
 - Provides priority inheritance configuration in the RTS
- Dynamic Priorities
- Time Critical Sections
- Abort Via Task Identifier
- Time Slicing

Scheduling Control

- ☞ Task Suspension
 - Provides a low-level means of controlling dispatching and execution, in a cooperative manner
- ☞ Two-Stage Task Suspension
 - Provides low-level controls that avoid race conditions
- ☞ Asynchronous Task Suspension
 - Allows one task to bilaterally prevent execution of another
- ☞ Synchronization Discipline
 - Allows specification of criteria for queuing entry calls and choosing among open select alternatives

Asynchronous Cooperation

- ☛ Resources
 - Provides efficient access control for (hardware) resources
- ☛ Events
 - Provides efficient task notification of latched conditions
- ☛ Pulses
 - Provides efficient task notification of non-latched conditions
- ☛ Buffers
 - Provides efficient asynchronous intertask communication
- ☛ Blackboards
 - Provides efficient inter-task messages

Asynchronous Cooperation

- Mutually Exclusive Access to Shared Data
- ☛ Broadcasts
 - Provides an efficient message broadcast capability
- ☛ Barriers
 - Allows simultaneous resumption of a fixed number of waiting tasks
- ☛ Asynchronous Transfer of Control
 - Supports ATC for fault recovery, mode changes etc.
- ☛ Shared Locks
 - Provides a very sophisticated lock facility
- ☛ Signals
 - Supersedes previous "Asynchronous Entry Call" interface

Interrupt Support

- Interrupt Management
- Trivial Entries
- Fast Interrupt Pragmas

Compiler Directives

- Pre-elaboration of Program Units
- ☞ Access Values That Designate Static Objects
 - Provides a more portable means to reference static objects
- ☞ Passive Task Pragmas
 - Provides a standardized approach to task "passification"
- ☞ Unchecked Subprogram Invocation
 - Provides more reliable means of invocation by address
- ☞ Data Synchronization Pragma
 - Allows CIFO task synchronization facilities to be used to share data

Memory Management

- Dynamic Storage Management
 - Provides predictable, flexible storage management facility

Deleted Entries

- Special Delays
 - One of the vendors convinced us that it was counter-productive
- Transmitting Task Identifiers Between Tasks
 - Removed in lieu of a more safe, coordinated approach to distribution facilities

Priority Inheritance

```
package Priority_Inheritance_Discipline is
  procedure Set_Priority_Inheritance_Criteria;
  procedure Reset_Priority_Inheritance_Criteria;
end Priority_Inheritance_Discipline;
```

```
pragma Set_Priority_Inheritance_Criteria;
```

- Enables/disables priority inheritance
- Procedural interface allows switching !!

Task Suspension

```
with Task_IDs;
package Task_Suspension is
  procedure Enable_Dispatching;
  procedure Disable_Dispatching;
  Function Dispatching_Enabled return Boolean;
  procedure Suspend_Self;
  procedure Resume_Task( Target : in Task_Ids.Task_Id );
end Task_Suspension;
```

- Tasks can control their own suspension
- Safe if not multiprocessing

Two-Stage Task Suspension

```

with Task_IDs;
package Two_Stage_Task_Suspension is
  Suspension_Error: exception;
  procedure Will_Suspend;
  procedure Suspend_Self;
  procedure Resume_Task( Target : in Task_Ids.Task_Id );
end Two_Stage_Task_Suspension;

```

- Safe for multiprocessing

Asynchronous Task Suspension

```

with Task_IDs;
package Asynchronous_Task_Holding is
  procedure Enable_Holding;
  procedure Disable_Holding;
  function Holding_Enabled return Boolean;
  procedure Hold_Task( T : in Task_Ids.Task_Id );
  procedure Hold_Task( T : in Task_Ids.Task_Id; Held : out Boolean );
  procedure Release_Task( Target: om Task_Ids.Task_Id );
end Asynchronous_Task_Holding;

```

- Controversial, but considered necessary
- All three Entries designed to interact predictably

Designating Static Objects

```
generic
  type Object is limited private;
  type Reference is access Object;
function Make_Access_Value( Static : Object ) return Reference;
pragma May_Make_Access_Value( <type_mark> );
Make_Access_Supported : constant Boolean := <value>;
```

CIFO Procurement Issues

- "More" is not "Better"
 - Unused Entries slow down others
 - Some Entries will never be implemented
 - Some Entries will conflict with others
- Conformance is only per Entry
 - Semantics of Entry
 - Interactions with other Entries

Adult Programming

- Don't mix conflicting Entries
 - Example: Various scheduling controls
 - Use the Interactions Matrix
 - Some Semantic ambiguities may still exist
- Don't use the "dubious" Entries
 - The procedural interface to Priority Inheritance Discipline (use the pragma)

CIFO and Ada9X

- Should 9X obviate a CIFO?
- A new "9X" version will eventually exist
 - Many Existing Entries will be removed
 - Some new Entries will be required!
- Ada83-based CIFO is needed now

Future Efforts

- Distributed Systems
- Multiprogramming
- Multi-level Security
- CIFO test suite for "conformance"
- Continuing refinement of existing Entries

Concluding Remarks

- Some controversial interfaces are defined
 - Low level asynchronous task control
- Some questionable interfaces are defined
 - Procedures that require paradigm shift at runtime
- Still the best approach available
 - Some proven, very useful interfaces are standardized
 - Meets the needs of the realtime community *now*
- Can serve as a bridge for Ada9X
 - Some interfaces are now in Ada9X, in one form or another
 - Education re: issues addressed by CIFO as intro to Ada9X

Other ARTEWG Activities

- Catalog of Runtime Implementation Dependencies
- Framework For Describing Ada Runtime Environments Model Runtime System Interface (MRTSI)
- Ada9X Revision Requests
- Ada9X Realtime facilities design review

Where to Get a Copy

- Send a self-addressed, postage-paid (\$3.00) envelope to

Mike Kamrad
Paramax Electronic Systems
M/S U1M30
PO Box 64525
St. Paul, MN 55164-0525



CMIT

Session VI

**REAL-TIME ACTIVITIES PETRO
CHEMICAL INDUSTRIES**

3:15 - 4:45

Session Leader:

Ernest Green, *Texaco Inc.*

Presenters:

Intelligent Alarming

W.B. Braden, *Texaco Incorporated*

Expert Systems in Process Industries

G. M. Stanley, *Gensym Corporation*

Fault Detection & Diagnosis using Neural Network Approaches

Mark A. Kramer, *Massachusetts Institute of Technology*



**MI
SAFETY CRITICAL
SYSTEMS
ION**

RESEARCH & APPLICATIONS

rics

SYMPOSIUM '92

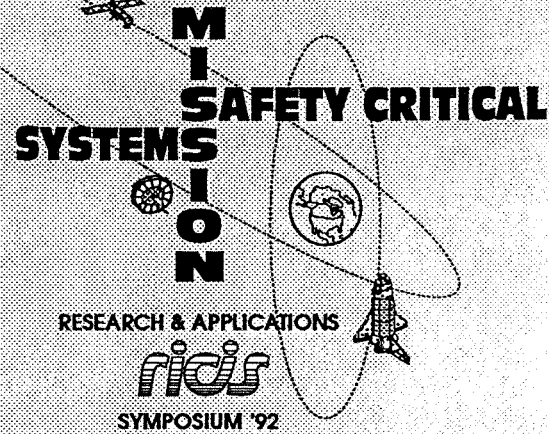
**REAL TIME ACTIVITIES
PETRO CHEMICAL INDUSTRIES**

Session Leader: Dr. Ernest Green

BIOGRAPHY

Dr. Green is Manager Research and Application in Texaco's Information Technology Department. His responsibilities include the management of teams engaged in research on the potential applications of artificial intelligence, operations research, and decision analysis in Texaco.

Dr. Green received his B. S. in Chemical Engineering from Mississippi State University in 1956 and his Ph.D. in Chemical Engineering from Rice University in 1965.



012 01
27242
P. 5

1995102254

324068

51

INTELLIGENT ALARMING

W. B. Braden

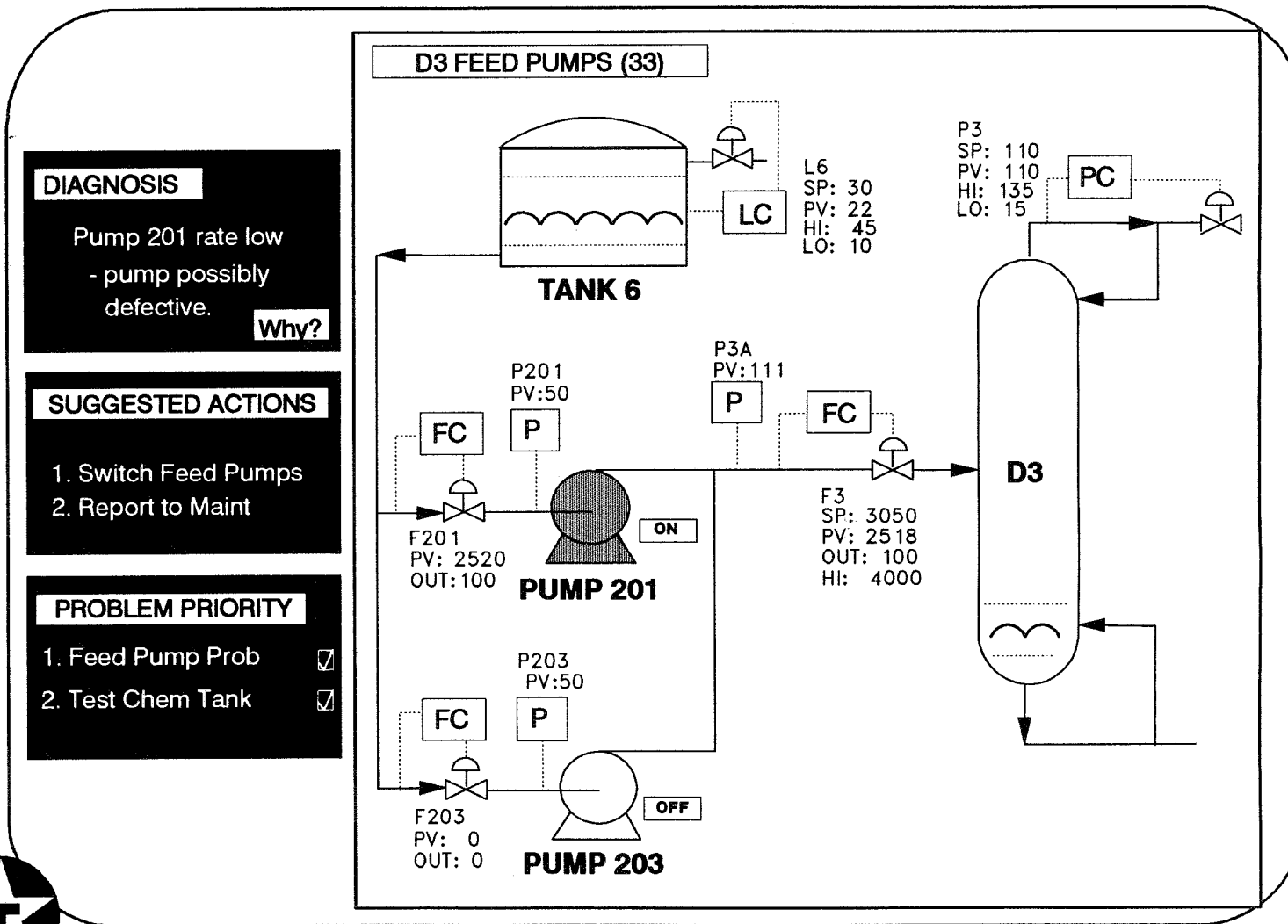
ABSTRACT

This talk discusses the importance of providing a process operator with concise information about a process fault including a root cause diagnosis of the problem, a suggested best action for correcting the fault, and prioritization of the problem set. A decision tree approach is used to illustrate one type of approach for determining the root cause of a problem. Fault detection in several different types of scenarios is addressed including pump malfunctions and pipeline leaks. The talk stresses the need for a good data rectification strategy and good process models along with a method for presenting the findings to the process operator in a focused and understandable way. A real-time expert system is discussed as an effective tool to help provide operators with this type of information. The use of expert systems in the analysis of actual vs predicted results from neural networks and other types of process models is discussed.

BIOGRAPHY

Bill Braden is a senior technologist in the Advanced Technology Group of Texaco's Information Technology Department, located in Houston, TX. His present focus is on the use of artificial intelligence in the areas of process alarming and control. Prior work involved chemicals research, fuels research, and tertiary oil recovery research.

Focused Advice



DIAGNOSIS

Pump 201 rate low
 - pump possibly defective.

Why?

SUGGESTED ACTIONS

1. Switch Feed Pumps
2. Report to Maint

PROBLEM PRIORITY

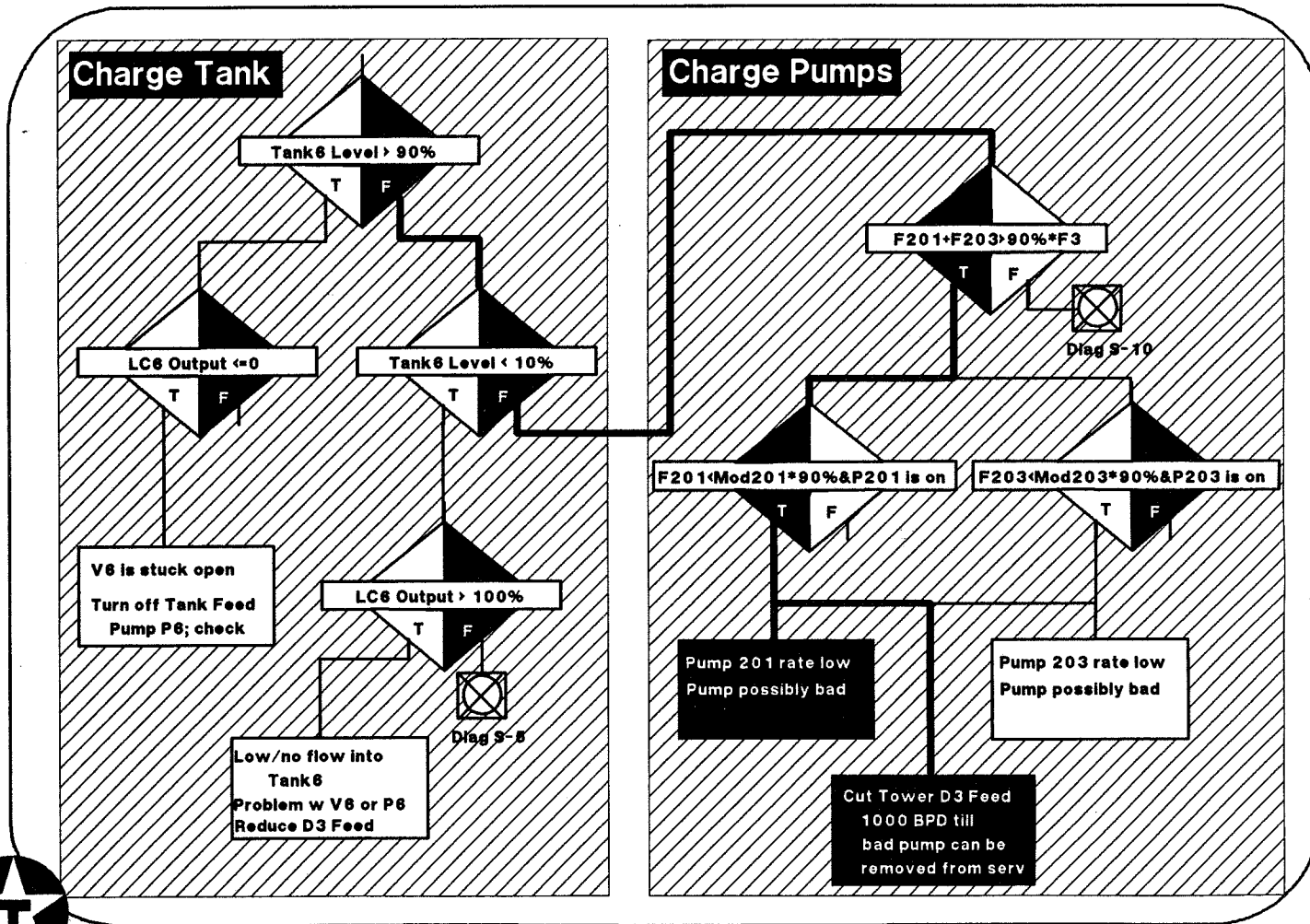
1. Feed Pump Prob
2. Test Chem Tank



TEXACO

WBB 8/14/92

Decision Tree

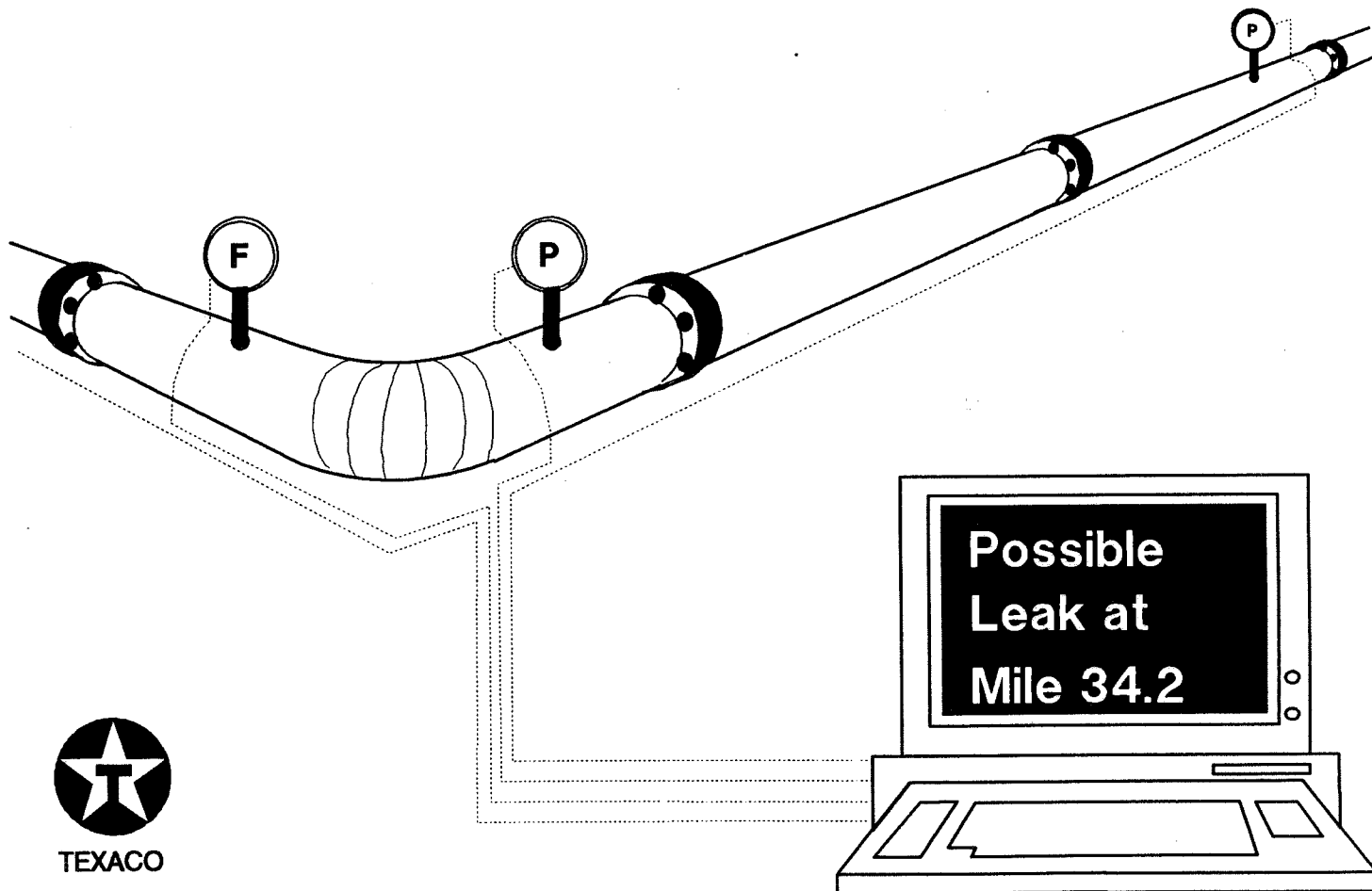


TEXACO

WBB 8/14/92

Neural Network Leak Detection

Safeguarding the Environment

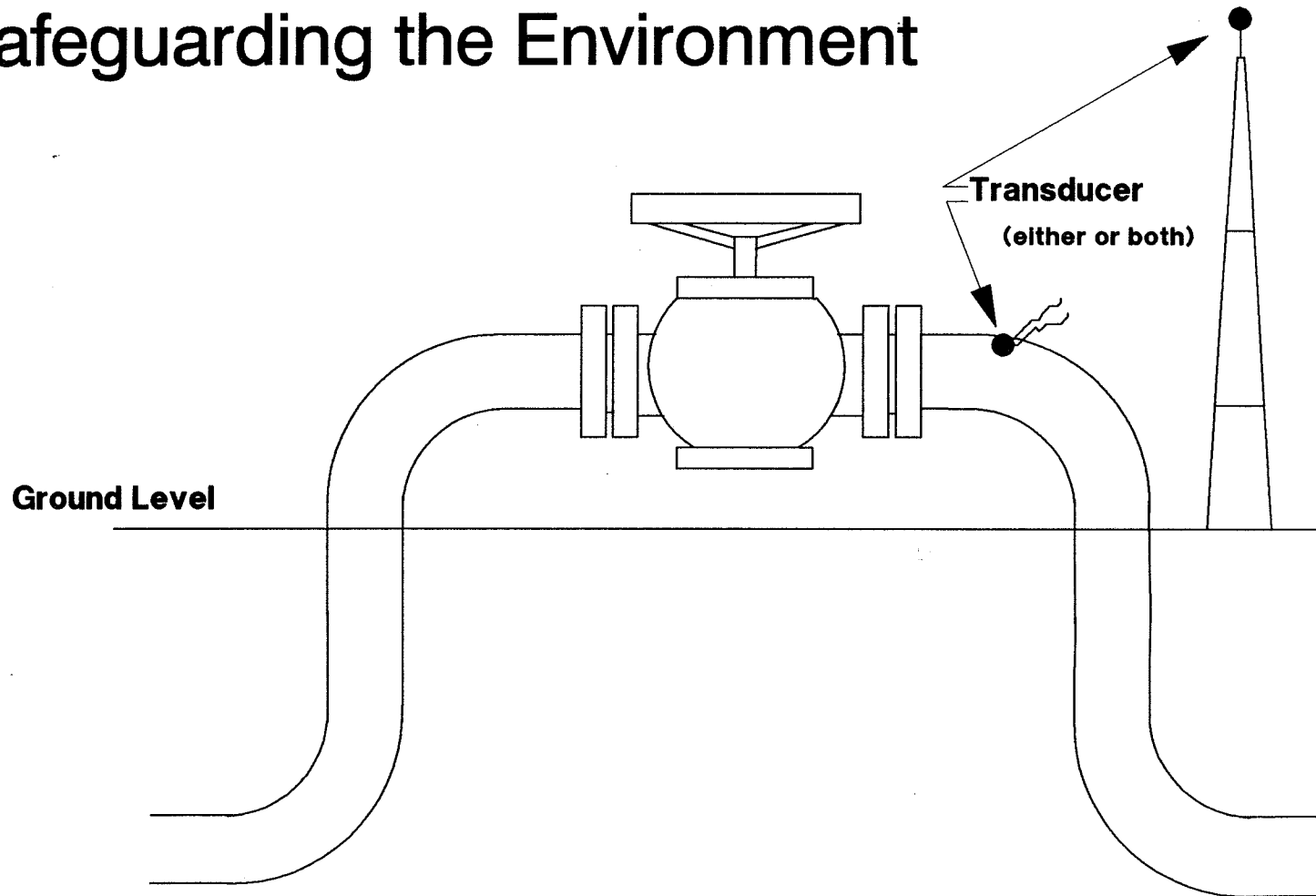


TEXACO

WBB 8/14/92

Sonic Leak Detection

Safeguarding the Environment



TEXACO

WBB 8/14/92



**MI
SAFETY CRITICAL
SYSTEMS
DIVISION**

RESEARCH & APPLICATIONS

ricis

SYMPOSIUM '92

27243

p 8

1995102955

324090

21

EXPERT SYSTEMS IN THE PROCESS INDUSTRIES

DR. G. M. Stanley

ABSTRACT

This paper gives an overview of industrial applications of real-time knowledge based expert systems (KBESs) in the process industries. After a brief overview of the features of a KBES useful in process applications, the general roles of KBESs are covered. A particular focus is diagnostic applications, one of the major application areas. Many applications are seen as an expansion of supervisory control. The lessons learned from numerous online applications are summarized.

BIOGRAPHY

Dr. Stanley, Principal Scientist at Gensym Corp., joined Gensym in 1987. He developed methodologies for practical application of real-time expert systems. He worked on and led several major diagnostics projects in the nuclear industry and in environmental controls. As Director of Applications Development, he led the development of new products such as the Diagnostic Assistant. He is the author of many technical papers in expert systems applications, data reconciliation, Kalman filtering and estimation theory, and simulation. He worked at Exxon Chemical from 1976-1987, holding key technical and management positions in process control, process engineering, optimization, expert systems, dynamic simulation, and information systems. Dr. Stanley completed his Chemical Engineering Ph.D degree at Northwestern University in 1976. His work in estimation and control advanced data reconciliation, Kalman filtering, and fault diagnosis, especially for network problems. He obtained his MS Ch.E. from Northwestern University, with a thesis on adaptive control. His BS ChE. was from Purdue University in 1971.

EXPERT SYSTEMS IN THE PROCESS INDUSTRIES

G.M. Stanley

Principal Scientist

Gensym Corporation - Southern Region, 10077 Grogan's Mill Road, Suite 100, The Woodlands, TX 77380

ABSTRACT

This paper gives an overview of industrial applications of real-time knowledge based expert systems (KBESs) in the process industries. After a brief overview of the features of a KBES useful in process applications, the general roles of KBESs are covered. A particular focus is diagnostic applications, one of the major application areas. Many applications are seen as an expansion of supervisory control. Finally, the lessons learned from numerous online applications are summarized.

BACKGROUND

Knowledge-based systems overview

Artificial Intelligence (AI) techniques include rule-based expert systems and object-oriented systems. The emphasis is declarative representation: separating the description (the knowledge) of a process, from the subsequent analysis of that knowledge by an inference engine. The knowledge is thus made more explicit, visible, and analyzable, instead of being hidden inside of procedural programming code. The knowledge is built as much as possible to be independent of the immediate application. Good expert system tools are generally based on an object-oriented paradigm, and we call them knowledge-based (KB) systems, or knowledge-based expert systems (KBESs).

A more detailed review of KBES applications in the process industries, with an emphasis on process control and a large number of references, is given by Stanley(1991). Descriptions of the features needed in a KBES for real-time control are given by Rowan(1989), Moore and others (1988), Moore, Stanley & Rosenof(1990), and Hoffman, Stanley & Hawkinson(1989).

A KBES in the process industry is often an extension of supervisory process control. Control technology generally emphasizes quantitative processing, while KBESs integrate both qualitative and quantitative processing. A KBES provides a general framework for integrating technologies as diverse as control design and operation, neural nets, rule-based systems, symbolic cause/effect models, logic networks, differential equation solving, and scheduling algorithms.

Some features of Knowledge-Based Expert Systems useful for online systems

Current online industrial applications are generally built within shells, which package a combination of tools. Different KBES shells may include some of the following features useful for online control applications:

- objects with attributes
- class hierarchy for objects, with inheritance of properties and behavior
- associative knowledge, relating objects in the form of connections and relations
- structural knowledge (e.g., "part-of" relation)
- representation and manipulation of objects and connections graphically
- rules and associated inference engine
- procedures
- analytic knowledge, such as functions, formulas, and differential equation simulation
- real-time features such as a task scheduler for concurrent operations, time stamping and validity intervals for variables, history-keeping, and data interfaces
- interactive development and run-time environment

Not all shells contain all these features. This paper is based mainly on experiences of users of G2, a real-time KBES shell which does include all these features.

The emphasis in a KBES is in building up descriptions, or knowledge, independent of the subsequent use of that knowledge in multiple applications. For instance, the developer specifies the types of objects in the plant, and specifies conditions which might correspond to a fault. The easy buildup of this declarative knowledge, combined with the available graphical interfaces, encourages a rapid prototyping and iterative refinement approach to software development.

Users often use a graphics-oriented KBESs to create a graphical language by defining the behaviors of objects and connections. For instance, a system based on AND and OR gates is a typical graphical language. Continuous control system engineers generally think in terms of data flow languages consisting of processing blocks and signals. Another common approach is to define objects representing actions, connected by directed arcs specifying sequential or concurrent execution. The GDA (Gensym Diagnostic Assistant) product, built using G2, is a complete graphical language encompassing both data flow (filters, AND & OR gates, etc.) and sequential control

In general, users of KBESs are representing almost everything as objects. It fits well with the way they think.

GENERAL ROLES OF A KBES IN THE PROCESS INDUSTRIES

An overview of KBES applications in the process industries, with an emphasis on process control, is given by Stanley(1991). That paper includes a number of case studies, and numerous references to successful applications illustrating the summarized points made in this paper. Some roles of expert systems in process control have been outlined by Stephanopolous (1990) and Árzén(1990). An overview of some current and expected applications is given by Rehbein and others (1990). Rosenof (1990) has summarized some roles for KBESs in batch process automation. Many of the online applications span more than one of the areas defined below, exploiting the usefulness of a KBES as a general framework:

The following are proven successful application areas for a KBES:

- Fault diagnosis: early detection, root cause analysis/alarm filtering, repetitive problem recognition, test planning, alarm management
- Supervisory control
 - Complex control schemes
 - Recovery from extreme conditions
 - Emergency shutdown
 - Heuristic optimization, e.g., debottlenecking
 - Startup or shutdown monitoring
 - Batch phase transition detection and subsequent control mode switching
 - Process and control performance monitoring
- Statistical Process Control (SPC) and subsequent assignment of causes
- Real Time Quality Management (combination of the above)
- Online "smart" operator and troubleshooting manual
- Sequential or batch control
- Control system validation
- Object-oriented simulation of processes and control systems

The following KBES application areas are actively being developed and tested by industry:

- Scheduling
- Operator training, with real-time simulation
- Tank farm management
- Formalizing compliance with ISO-9000 (quality), government, or other standards

Some evolving and future roles of a KBES in the process industries:

- Predictive maintenance
- Process validation
- Intelligent supervision of adaptive control, model identification, parameter estimation, state estimation, data reconciliation, optimization, neural network training & run-time coordination
- Automated design of control systems (and implementation)

Economic justification for a KBES in the process industries

After some early experimentation, applications are now generally justified based on economics as well as safety. Some companies have published information quantifying substantial benefits. For instance, DuPont has stated that they "routinely" see returns on investment as high as 10 to 1 (Rehbein and others, 1990; Rowan, 1989). Monsanto's evaluation of its first online system showed benefits of \$250K/year (Mertz, 1990; Spang Robinson Report, 1989, Rehbein and others, 1990). Other examples can be found in the review paper by Stanley (1991). The justifications for a typical application such as diagnosis, include:

- Safety
- Real-time quality management/quality control
 - Early problem detection with multiple variables
 - Determining the assignable causes of problems
- Yield/production
- Loss prevention
- Equipment protection
- Environmental Protection
- Sensor/model validation for successful plant optimization
- Formalizing compliance with ISO-9000 (quality), government, or other standards

GENERAL LESSONS LEARNED

Real-time KBESs are robust enough to have succeeded in numerous applications, including closed loop control.

Some current systems operate in a "closed loop" mode, manipulating valves or controller setpoints. Many of the current systems already occupy a "grey" area between open and closed loop control: the control goal is closed-loop, but an operator is in the feedback loop, routinely approving the recommendations. Many of the open-loop applications will migrate to closed loop, as people build up confidence.

Significant benefits are derived in areas complementary to conventional controls, such as diagnosis, quality management, and abnormal operation

Significant benefits have been achieved. Many of the credits are in the same areas as good process control, e.g., process repeatability, quality improvement, achieving best demonstratable operation, shorter batch time, lower waste or energy costs, and avoidance of accidents. However, the reasons for the benefits often complement those of process control, since they are often derived during periods of unusual operation, or from better planning of the normal control operations. KBES-based diagnostics are needed as part of the overall control system to catch major problems such as sensor failure, and then disable the fragile, "normal" control systems which only handle normal operations. Quality problems can be thought of as faults -- they are economic faults, just less severe than safety problems. Diagnostic techniques typically also are used in batch control systems to detect or plan the transition from one operating phase to another.

Normal models and controls break down during the extreme operation. There, the more effective models or actions under extreme conditions are likely to be simpler, based on first principles or on heuristics. These alternate controls are easier to build in a KBES than in conventional systems.

KBESs complement SPC techniques by earlier problem detection and determining assignable causes, achieving Real-Time Quality Management

SPC tools are sensitive detectors of problems. However, they offer little or no guidance as to the root causes (assignable causes) of problems, or how to correct the problems. This is a fundamental limitation, because standard SPC techniques do not capture process model knowledge and use it. A KBES can apply SPC to detect problems, and then pinpoint the cause of the problems. Thus, the broader problem of "maintaining product quality" can be addressed through a combination of SPC techniques, diagnostic techniques, and conventional control systems during normal operation. This broader approach to "Real Time Quality Management" has been successfully applied by DuPont and others.

Pure SPC systems also require the users to wait until faults have propagated and repeatedly caused off-spec products. By building in process knowledge, faults can be detected and corrected long before SPC techniques recognize a product problem. Diagnostic techniques implemented in a KBES can use SPC techniques as sensitive detectors of problems, but also provide a broader framework for building in the knowledge to determine the causes of problems and correct them. For example, if a valve sticks, a reflux drum may empty, ending reflux to a distillation tower, causing product to go out of specification half an hour after the fault, with detection at an online analyzer within another 10 minutes, and confirmation from the laboratory in 2 hours. Diagnostics could detect the stuck valve in less than a minute.

KBESs return significant economic benefits

The systems can now be justified for economic reasons, in addition to safety.

Significant benefits are derived from productivity in development

While the earliest expert systems were major efforts, a graphics-oriented real-time KBES now can significantly shorten development time vs. conventional coding. The ability to rapidly prototype and get user input is a major benefit. While any of these systems could be implemented in conventional code, it would be difficult, more time-consuming and error-prone, and harder to maintain.

KBESs reduce the gaps between specification, implementation, and run-time

KBESs encourage declarative representation of the information needed for design of a system, such as objects with attributes which are used to build models. The process schematic itself is part of the design basis, and can be used directly at run-time. The design procedure itself can be automated. For instance, goals and subgoals can be represented as objects suitable for deriving control strategies. Domain-specific heuristics on selection of controlled and manipulated variables can be explicitly represented as rules or objects.

In an integrated package, many of the objects (such as the process schematic) used by the designer can be used by the end user. Status indications via color are useful to both the designer and end-user. A programmer separate from the designer and end user is generally not needed. A separate software package for design and run-time use are not needed.

Maintainability is a major issue

Early custom-coded systems were not maintainable, and are no longer used. Maintenance is a major issue at plants, because they are always being modified, and related computer systems need to evolve with it. Modern KBES shells provide a better framework. Systems must be changeable in a natural way by the users, not just AI developers.

System integration is a major issue

A significant portion of the overall effort is in systems integration. Tools which build in extensive support for real-time data interfacing save significant development effort.

Graphics-oriented KBESs are an integrating technology

Due to their high-level ability to represent, manipulate, and display knowledge in various forms, graphics-oriented KBESs can be used as a tool to integrate other techniques. One KB representation

can be used for multiple purposes. Work is under way at various locations using a KBES to integrate such diverse technologies as neural networks, fault trees, databases, and expert system rules.

A KBES can fill the "CIM gap" between process control and planning & scheduling. For instance, once the KBES has a representation of the plant schematic, the recipes, and the processing sequence and estimated processing times, that same representation can be used both for planning purposes, and then to carry out the sequential control. The key is that the plant and product information is represented in a way independent of the application. In a continuous plant, a hybrid system can decide when it is time to do an emergency shutdowns, and carry out the shutdown. In a batch process, the hybrid system can detect the end of one phase of operation, and switch control schemes for the next phase of operation.

KBESs specialized for real-time use are needed for process control applications

Earlier attempts to extend the traditional static expert system shells, or to code a system from the beginning, were generally interesting learning experiences. These mostly ineffective attempts were generally driven periodically by batches of data placed in files.

However, for the dynamic industrial environment, these approaches generally proved too slow, too difficult to be economically justified or maintainable, and often too unreliable. A specialized real-time KBES uses an asynchronous processing model for data acquisition and task execution within the expert system. The necessary features for history-keeping, time stamping, and so on, are provided. Also, early LISP-based systems, without special memory-management provisions to prevent garbage collection, could suffer seemingly-random pauses during garbage collection (memory reclamation), unacceptable for real-time operation. A real-time KBES should not garbage collect at run time.

LESSONS LEARNED IN KNOWLEDGE REPRESENTATION

Graphical specification of knowledge is effective

Users like developing graphical problem-specification languages. In many cases, users have defined their own graphical languages, using blocks to represent numerical or logical operations, and to represent sequential action steps. Based on these experiences, Gensym has built the GDA product already mentioned. Many users have made extensive use of the information available directly from process schematics built out of objects with connections between them.

Generic knowledge libraries shorten development time

Many users are building libraries which can be reapplied at different sites, based on analyzing a process schematic. This is especially applicable in diagnostics, where low-level failures in valves and sensors are essentially the same in all plants. This results in rapid transfer of technology and development, uniformity, and maintainability. The knowledge libraries speed applications at the first site as well, because much of the configuration for the entire site is for repetitive elements such as valves and controllers.

Symbolic and numerical filtering, and evidence combination techniques for managing noise and uncertainty are important

Event and trend detection, with their associated upstream models and filtering, provide the interface between the continuous, external world, and the higher-level, usually symbolic states in the knowledge based system. Noisy signals can lead to transient false conclusions, obscure the true conclusions, and lead to excessive forward chaining.

To reduce the impact of noise, you can filter heavily and accept the resulting lags in many cases, since a fast feedback loop is not within the application. Nonlinear techniques such as various forms of hysteresis based on state or time can be extremely useful. The importance of filtering of various types has been reported for most of the applications. In addition, SPC techniques are now being thought of as a form of filtering for input to the rest of the expert system as well.

In addition to conventional filtering, other techniques can be quite useful. Conversion from numerical values to symbolic values of "high", "low", or "OK", significantly reduces the number of state changes in subsequent processing (forward chaining). Various symbolic forms of filtering, such as latching and event counting have a significant role to play as well. Some users found it necessary to delay fault alarms until the condition had been true for a period based on time or event counts.

Just as errors can be reduced by combining multiple sensor values using numerical models (such as Data Reconciliation or Kalman Filtering), sensor evidence can be combined using techniques more powerful than simple discrete logic (true/false values only). Industry has only begun to experiment with various models of evidence combination and fuzzy logic, which can also help address these problems. These techniques will become more prominent in future applications. Users have indicated a desire for ranked lists of possible faults, which requires better schemes than discrete logic.

Quantitative information and models are often needed

A significant amount of knowledge has been abstracted by engineers into mathematical models. The best systems are hybrids of qualitative and quantitative techniques. This is intuitive, because the system is taking advantage of more knowledge about the process. Furthermore, in many of these systems, the simulation is specified in an object-oriented form. The user often creates graphical objects with attributes, and the library equations directly derive the necessary mathematical representation from that structure

Diagnostic systems based on deviations from quantitative models tend to be very sensitive to faults of all types, even when operation is close to normal. Model-based approaches can significantly increase sensitivity to real faults. Also, the time to recognize those faults is shortened, because they are detected within the normal operating range, before significant harm is caused by the fault.

Approaches based on deviations from models (residuals) also have the advantage of detecting some faults which were not even anticipated, but which affect the variables in the model equations. (In that case, the system can alert the operators, although not necessarily identify the exact cause). A good example is material and energy balance equations which do not explicitly account for an actual leak or pipe break, since they are low-probability events. However, if a pipe does break, the resulting large material and energy balance equation residuals will quickly indicate a problem, even if the logic does not explicitly derive a specific conclusion beyond the initial problem detection. However, minor problems such as mild sensor drift, mild process upsets, slightly larger than normal noise, or modelling error can all lead to incorrect detection of faults.

Most useful industrial systems involving continuous variables are hybrids of the model-based and pure symbolic approaches. The models can generate residuals, which then feed into the symbolic logic.

Knowledge-based systems provide good repositories for process technology, improving the uniformity of operator responses

Since the embedded knowledge is visible to the operators, is testable and generally can be queried, the operators can use it as a learning aid, and can continue to refine it. Whether the operators take manual actions based on the system, or allow the system to directly manipulate the process, the results are higher uniformity of control actions.

CONCLUSIONS

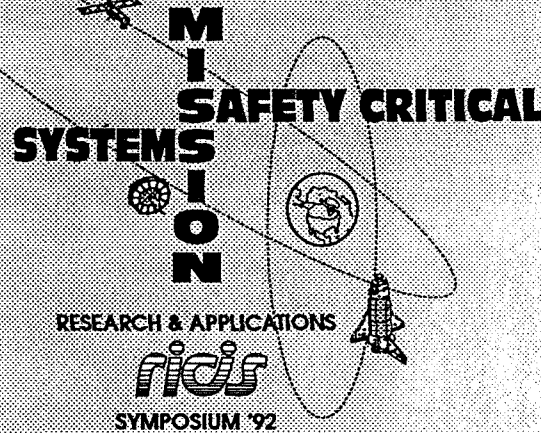
Online KBESs are making significant contributions to process control and management. They are economically justified. The applications and benefits are often in areas which complement traditional

process control technology, for instance, in handling abnormal situations, and in overall quality management. The KBES integrates new techniques with conventional controls.

Many lessons have been learned from the industrial experiences, such as the importance of filtering, the importance of integrating SPC tools, and the need for integration of quantitative models.

REFERENCES

- Årzén, K.-E. (1990). Knowledge-Based Control Systems -- Aspects on the Unification of Conventional Control Systems and Knowledge-Based Systems. *Proc. American Automatic Control Conference (ACC-90), San Diego, CA, 2233-2238.*
- Hofmann, A.G., G.M. Stanley, and L.B. Hawkinson (1989). Object-Oriented Models and Their Application in Real-Time Expert Systems. *Proc. Society for Computer Simulation International Conference, San Diego.*
- Finch, F.E., G.M. Stanley, and S.P. Fraleigh (1991). Using the G2 Diagnostic Assistant for Real-Time Fault Diagnosis. *European Conference on Industrial Applications of Knowledge-Based Diagnosis, Segrate (Milan), Italy, October, 1991.*
- Fraleigh, S.P., F.E. Finch, and G.M. Stanley (1992). Integrating Dataflow and Sequential Control in a Graphical Diagnostic Language. *Proceedings of the International Federation of Automatic Control (IFAC) Symposium on Online Fault Detection and Supervision in the Chemical Process Industries, Newark, DE, April, 1992.*
- Mertz, G.E. (1990). Application of a Real-Time Expert System to a Monsanto Process Unit. *Proc. Chemical Manufacturer's Association Process Control Conference, Miami, Florida, March, 1990.*
- Moore R. and others (1988). The G2 Real-Time Expert System. *Proc. Instrument Society of America (ISA-88), 1625-1633.*
- Moore, R., H. Rosenof, and G. M. Stanley (1990). Process Control Using a Real-Time Expert System. *Proc. 11th IFAC World Congress, Tallinn, Estonia, USSR, 234-239.*
- Rehbein, D., D. Deitz, S. Thorp, and L. Tonn (1990). Expert Systems in Process Control, *Proc. ISA, New Orleans, Louisiana, USA, Oct. 14-18, 1990, 1989-1995.*
- Rosenof, H.P. (1990). Expert Systems Come to Batch Process Automation. *Proc. Instrument Society of America International Conference (ISA-90), New Orleans, Louisiana, USA, 1367-1376.*
- Rowan, D.A. (1989). On-Line Expert Systems in the Process Industries. *AI Expert, August, 1989, 30-38.*
- Spang Robinson Report (1989). Real-Time Application Profile: Monsanto's Process Control System. *The Spang Robinson Report on Artificial Intelligence, Nov., 1989.*
- Stanley, G.M. (1991). Experiences using knowledge-based reasoning in online control systems. *International Federation of Automatic Control (IFAC) Symposium on Computer Aided Design in Control Systems, July 15-17, 1991, Swansea, UK.*
- Stanley, G.M., F.E. Finch, and S.P. Fraleigh (1991). An Object-Oriented Graphical Language and Environment for Real-Time Fault Diagnosis, *Proc. European Symposium on Computer Applications in Chemical Engineering (COPE-91), Barcelona, Spain, Oct. 14-16, 1991.*
- Stephanopoulos, G. (1990). Artificial Intelligence ... What Will Its Contributions Be to Process Control? in *The Second Shell Process Control Workshop, ed. David M. Prett and others, Butterworth Publishers, Stoneham, MA, USA, pp. 591-646.*



27244
P-31
1995 107756
3 27072
528

FAULT DETECTION & DIAGNOSIS USING NEURAL NETWORK APPROACHES

Prof. Mark A. Kramer

ABSTRACT

Neural networks can be used to detect and identify abnormalities real-time process data. Two basic approaches can be used, the first based on training networks using data representing both normal and abnormal modes of process behavior, and the second based on statistical characterization of the normal mode only. Given data representative of process faults, radial basis function networks can effectively identify failures. This approach is often limited by the lack of fault data, but can be facilitated by process simulation. The second approach employs elliptical and radial basis function neural networks and other models to learn the statistical distributions of process observables under normal conditions. Analytical models of failure modes can then be applied in combination with the neural network models to identify faults. Special methods can be applied to compensate for sensor failures, to produce real-time estimation of missing or failed sensors based on the correlations codified in the neural network.

BIOGRAPHY

Mark A. Kramer is currently Associate Professor of Chemical Engineering at the Massachusetts Institute of Technology. Professor Kramer received a Bachelor's degree at the University of Michigan in 1979 and a Ph.D. degree from Princeton University in 1983. He also has served as the Associate Director of the Laboratory for Intelligent Systems in Process Engineering at MIT. Professor Kramer has published over 45 papers in the areas of artificial intelligence in process engineering and neural networks.

Fault Detection & Diagnosis using Neural Network Approaches

Mark A. Kramer
Massachusetts Institute of Technology
Department of Chemical Engineering

Mission Safety Critical Systems: Research & Applications
Research Institute for Computing and Information Systems
University of Houston-Clear Lake
October 29, 1992

How to use neural networks to:

- 1) *detect*
- 2) *identify*
- 3) *rectify*

faults in processes and associated sensors.

Using neural nets involves **learning from examples.**

Two approaches:

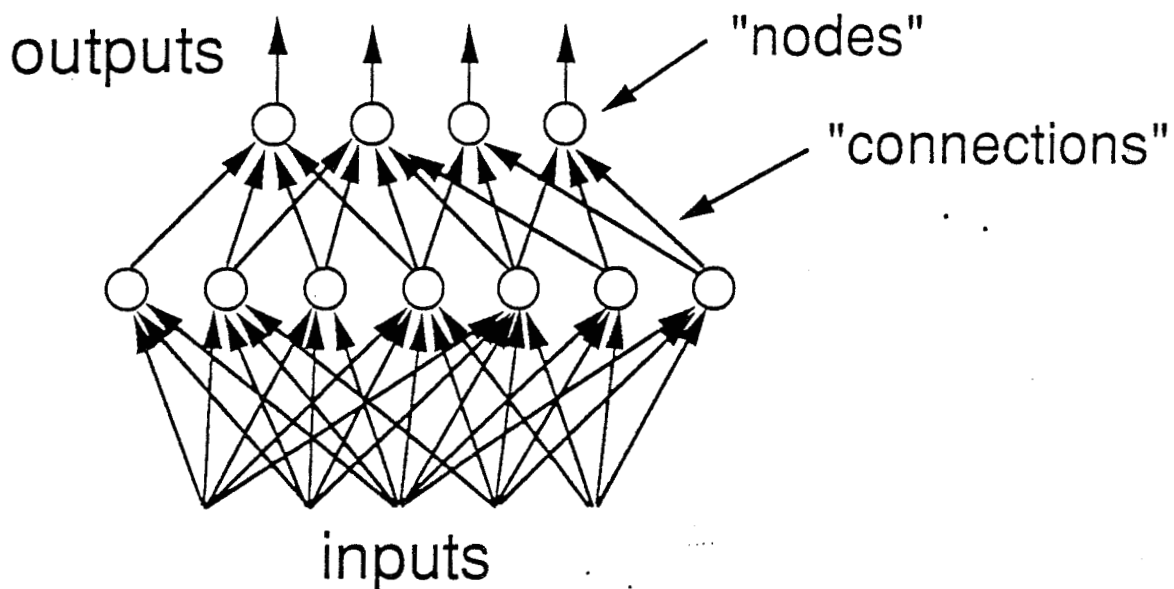
- 1) Learning with examples of normal and abnormal behavior
- 2) Learning with examples of normal behavior only

Useful in the absence of a **functional theory** of device behavior.

REVIEW OF NEURAL NETWORKS

Artificial neural networks are loosely based on how the brain carries out its low-level computations.

- Simple computational elements acting in parallel
- Neurons "fire" when excited by other neurons
- Capable of learning and responding differently to different input patterns



Input/output behavior determined by:

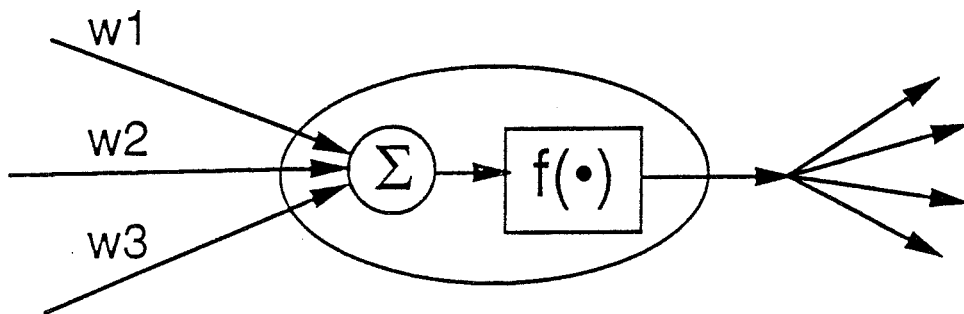
- Topology of network
- Computation of each neuron
- Adjustable parameters of connections and nodes

Two of the most important types of networks are:

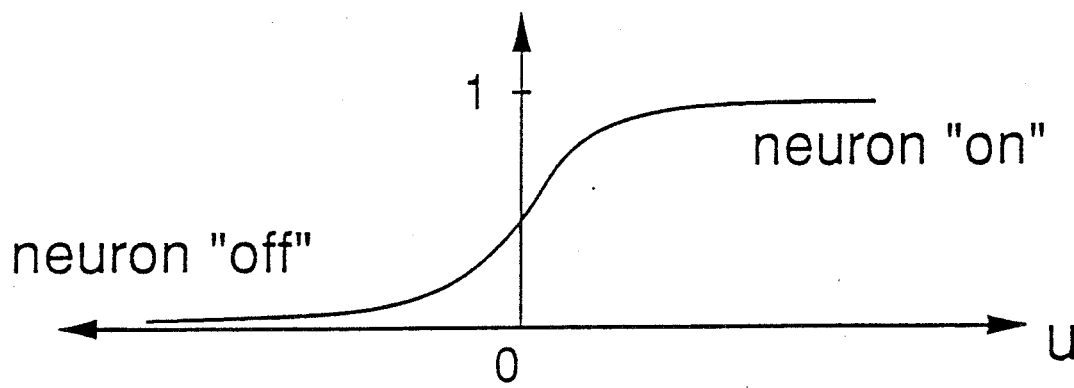
- **backpropagation networks (BPNs),**
- **radial basis function networks (RBFNs)**

Computation in Backpropagation Networks

- Layered architecture, usually input layer, output layer, plus one intermediate "hidden" layer
- Connections between nodes are weighted. Weights multiply the signal on the connection.
- Each node sums its inputs and then passes the result through a **sigmoidal** nonlinearity

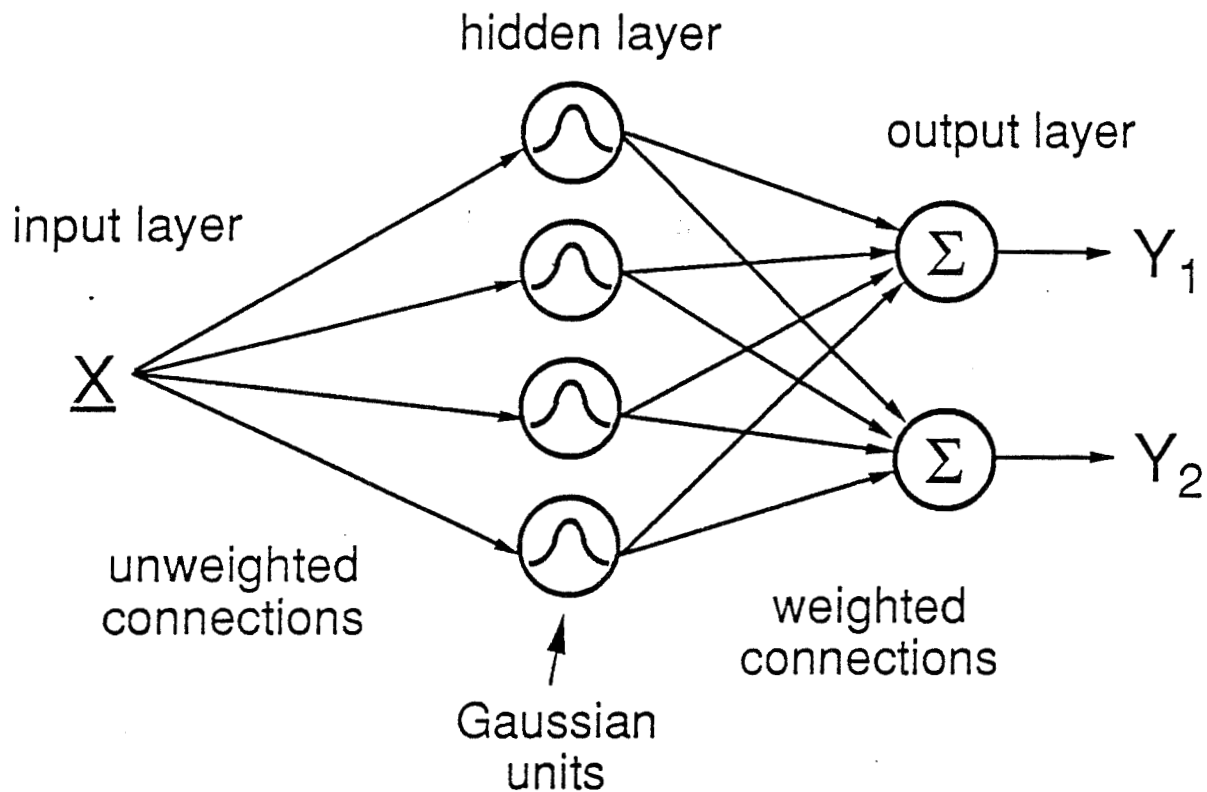


Typical sigmoid function: $f(u) = 1/(1+\exp(-u))$



Computation in Radial Basis Function Networks

- Similar to BPN but uses **Gaussian** nonlinearity in nodes
- Input/hidden layer connections not weighted, simply pass input vector \underline{X} to hidden layer



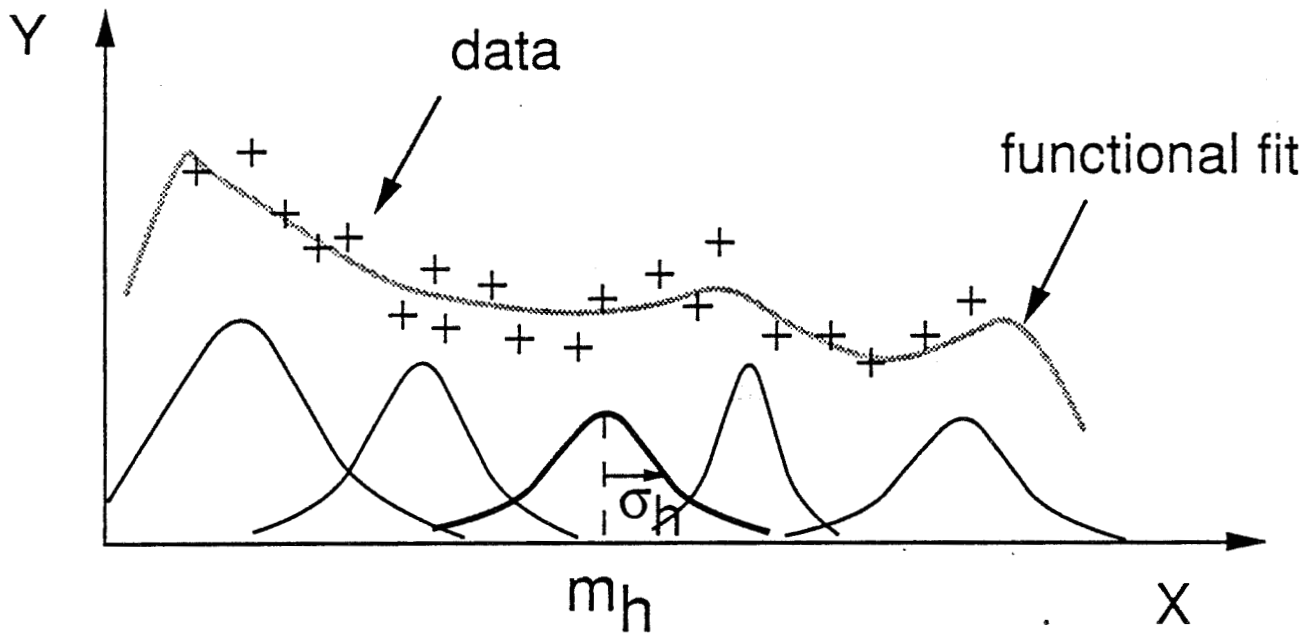
Each Gaussian unit has internal parameters representing a "unit center" \underline{m} and a "receptive width" σ

Output of Gaussian unit i , a_i , is based on the distance between inputs \underline{x} and the unit center \underline{m} :

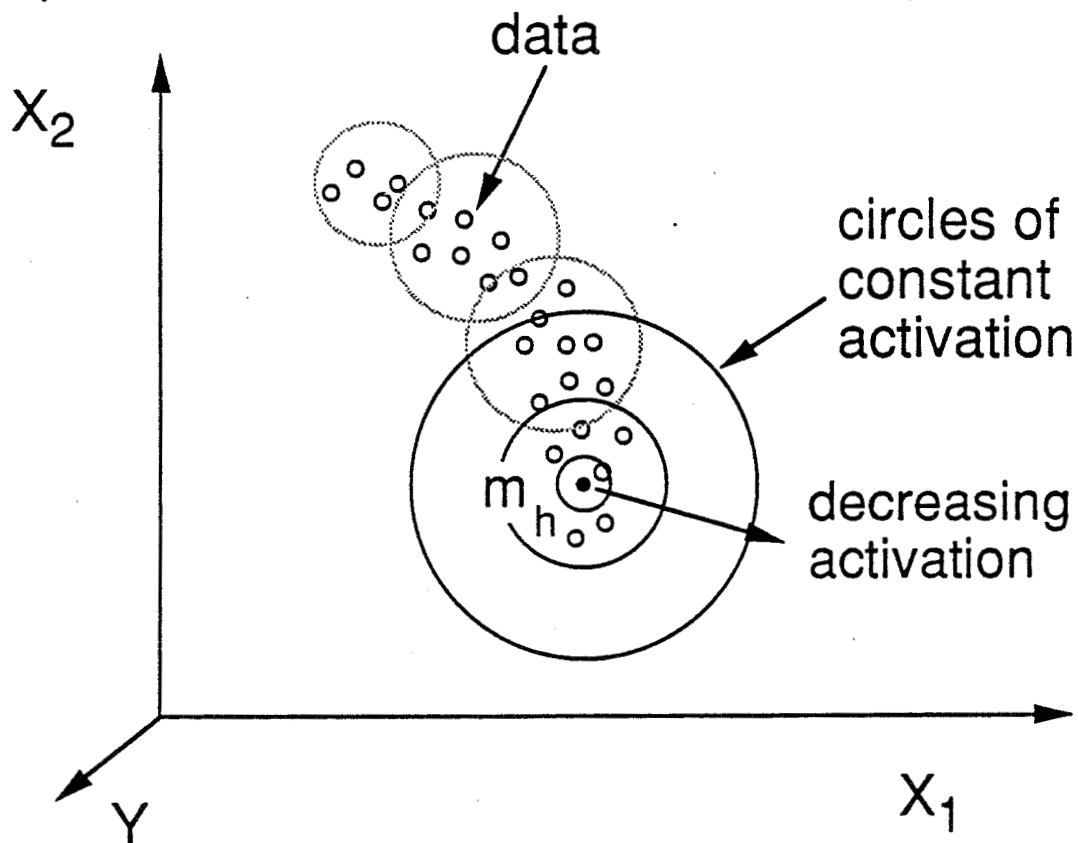
$$a_i = \exp\left(\frac{-\|\underline{x} - \underline{m}_i\|^2}{\sigma_i^2}\right)$$

Graphical interpretation of hidden nodes of RBFN:

One input dimension:



Multiple dimensions:



NEURAL NETWORK LEARNING

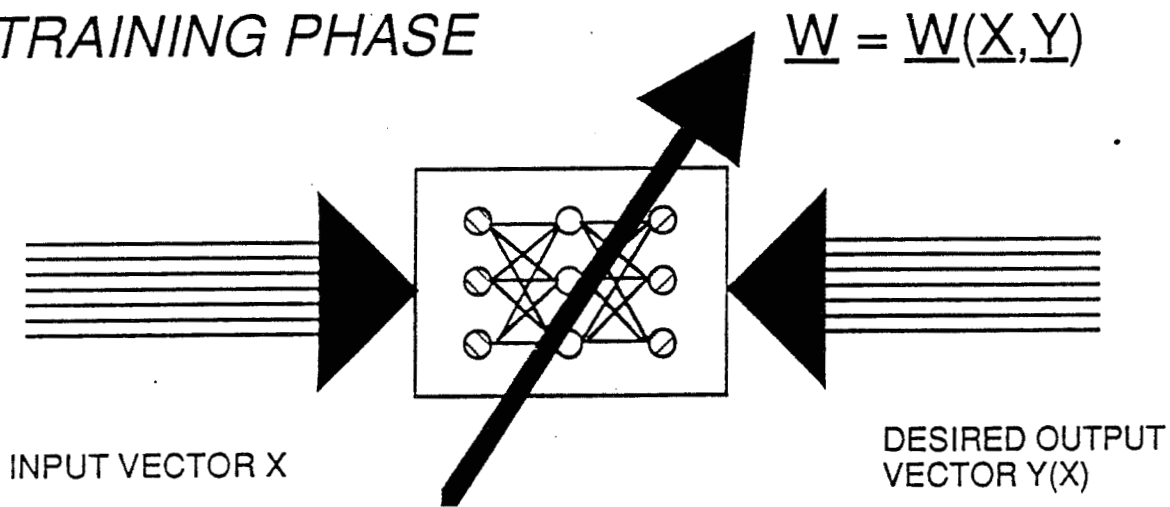
Takes place through an optimization of internal connection weights W.

A set of examples of desired input/output behavior $(\underline{x}, \underline{y})_i, i=1, \dots, K$ is required.

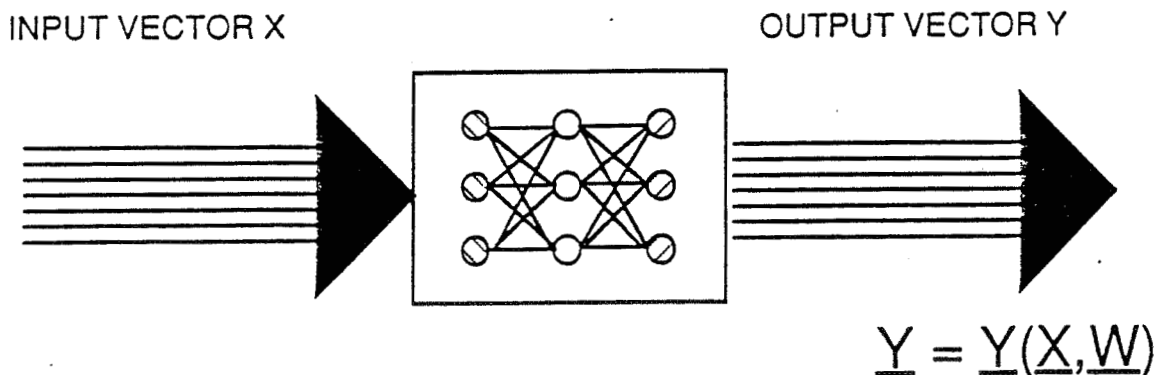
A **least-squares** fit is sought:

$$\min_w \sum_{i=1}^K [y_i - \text{Net}(x_i)]^2$$

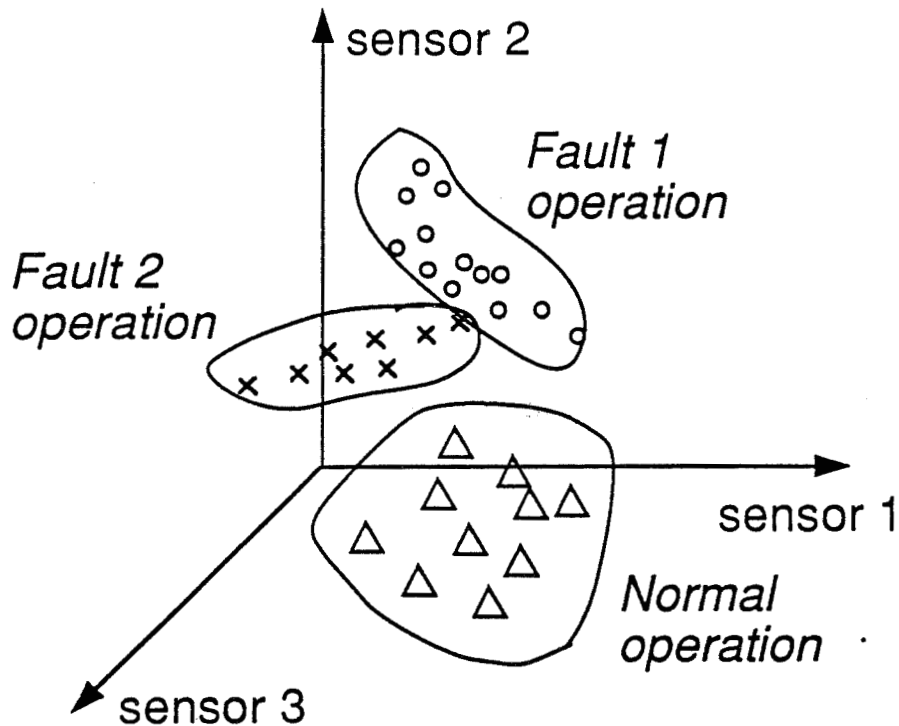
TRAINING PHASE



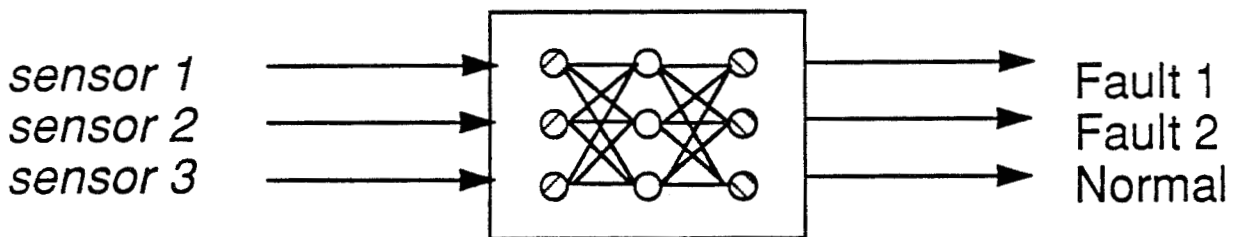
APPLICATION PHASE



Multi-Class Diagnosis Using Neural Networks



Neural network is capable of identifying more complex class regions than "high-low-normal"-style rules



NEURAL NETWORK

Network Training For Multiclass Diagnosis

<u>observables x</u>	<u>diagnosis</u>	<u>y</u>
{0.032, 0.099, -0.039}	→ fault 1	→ {1, 0, 0}
{0.016, -0.53, -0.465}	→ fault 2	→ {0, 1, 0}
{0.466, 0.022, -0.405}	→ fault 3	→ {0, 0, 1}

Example inputs x:

Feedstock characterization: {sp. grav., bubble pt., visc.,...}

Sensor data: {meas 1, meas 2,....}

Time series: {meas(t), meas(t-1), meas(t-2), ...}

Using least squares objective function, assuming:

- 1) Sufficient # of training examples
- 2) Examples in proportion to prior probabilities
- 3) Adequate network representational capacity

Then:

$$y_i = \frac{P(\text{fault } i \mid x)}{\sum_j P(\text{fault } j \mid x)} = \begin{array}{l} \text{relative} \\ \text{probability} \\ \text{of fault } i \end{array}$$

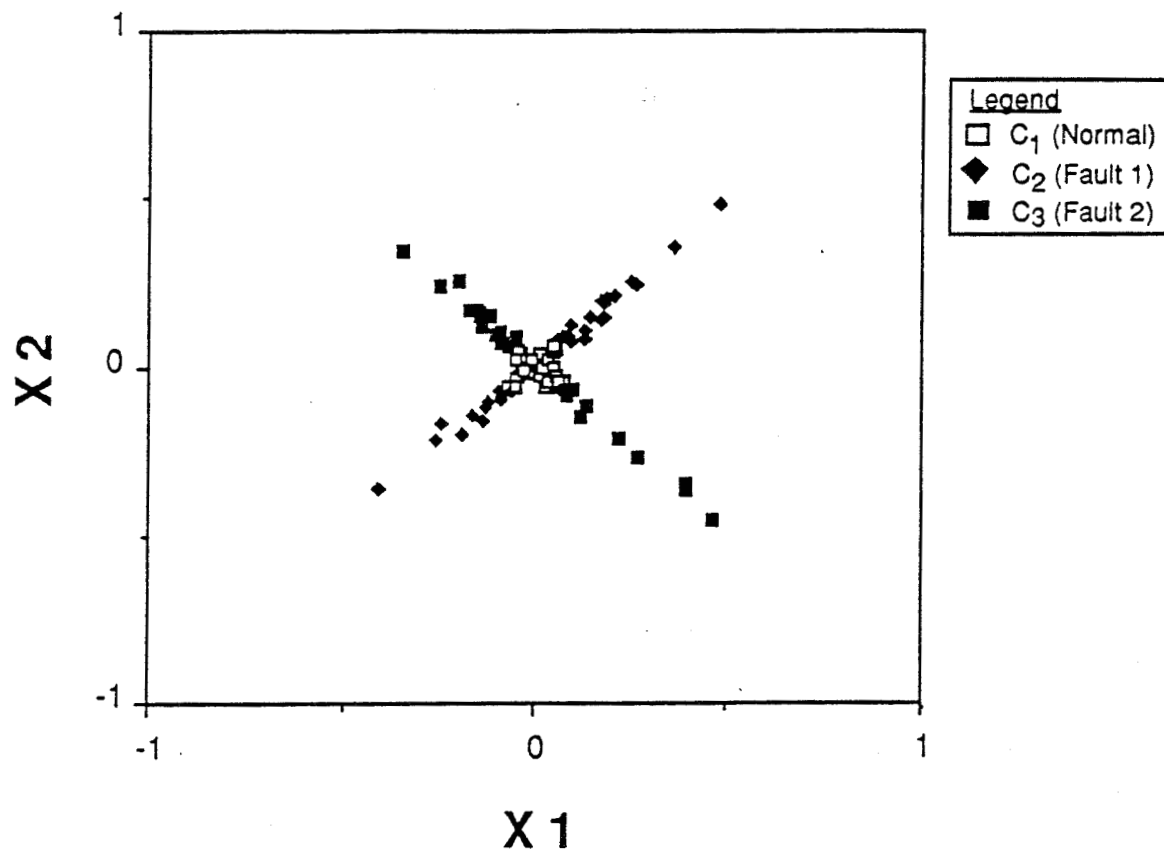
$\sum y_i \neq 1$ implies an invalid classification

RADIAL BASIS FUNCTION NETWORKS ARE
BETTER FOR DIAGNOSTIC PROBLEMS THAN
BACKPROPAGATION (SIGMOIDAL) NETS

Fault diagnosis example problem

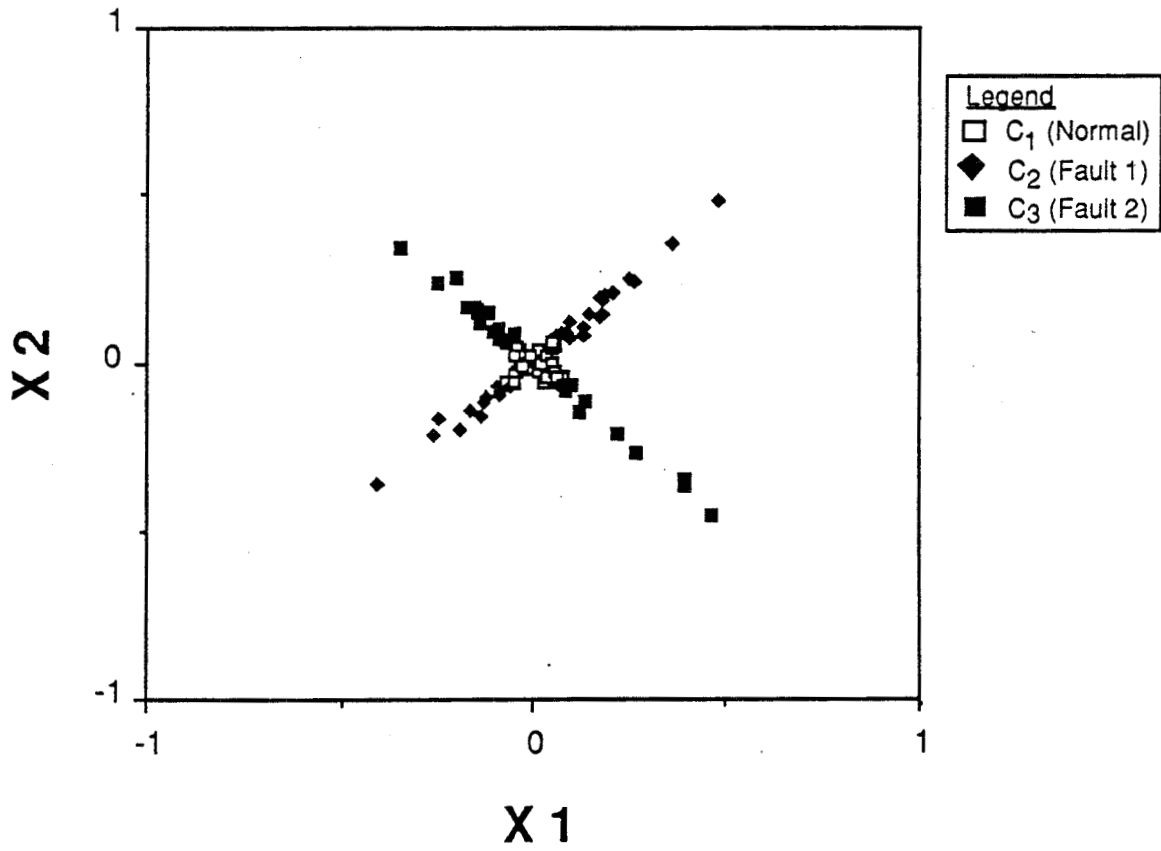
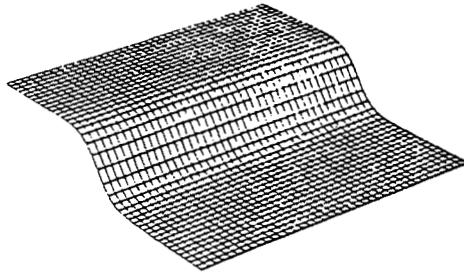
(see Kramer & Leonard, IEEE Control Systems 11, 31, April 1991)

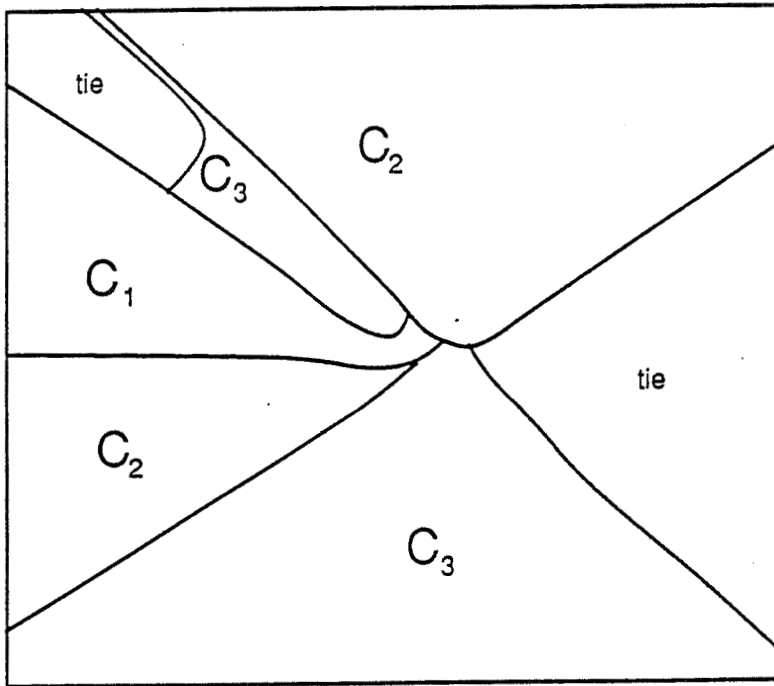
3 classes, 2 input dimensions
30 training examples of each class.



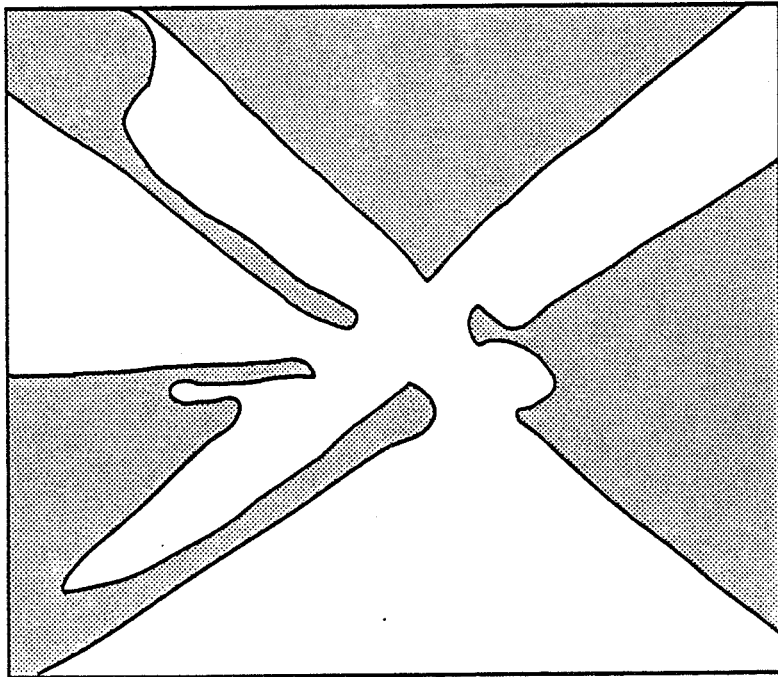
Backpropagation networks (sigmoidal nodes):

- Class regions divided with hyperplanes
- Tends to place class boundaries near "edge" of class
- Check sum $\sum Y_i \neq 1$ indicates some regions of insufficient training data (sufficient but not necessary)





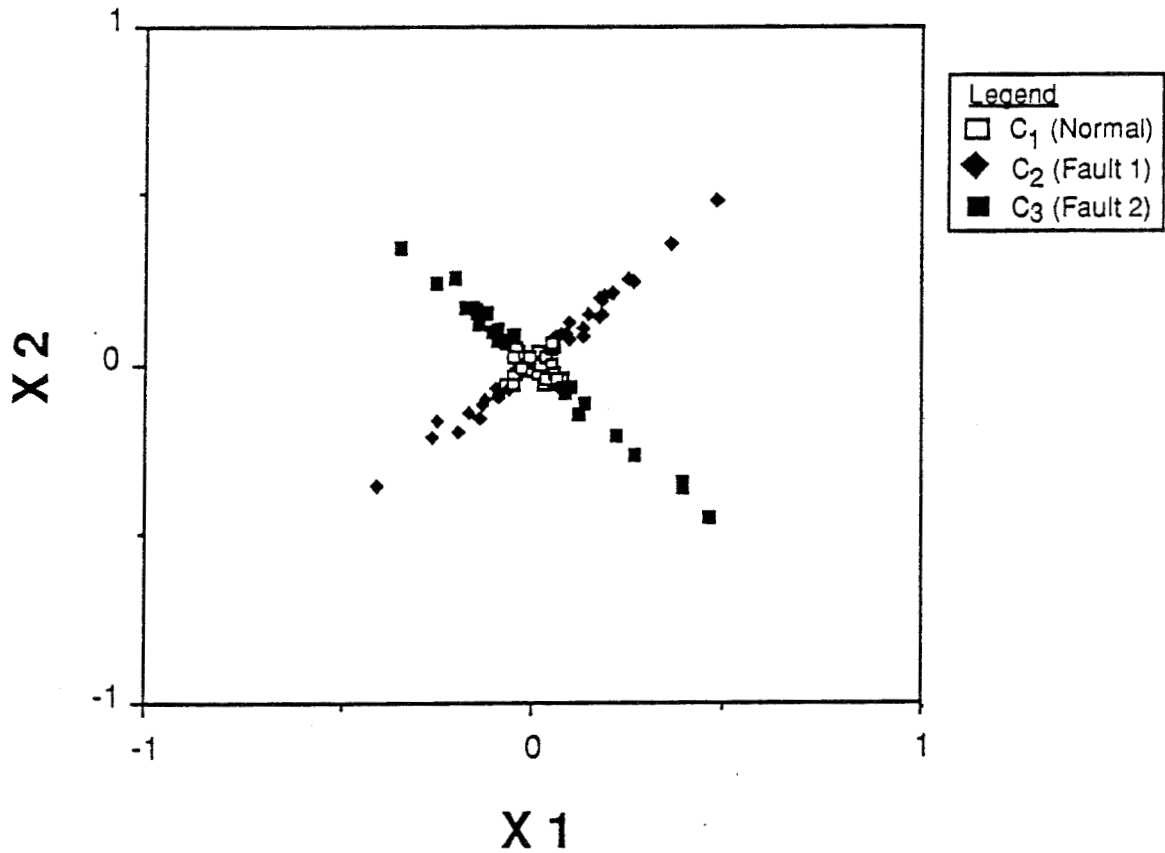
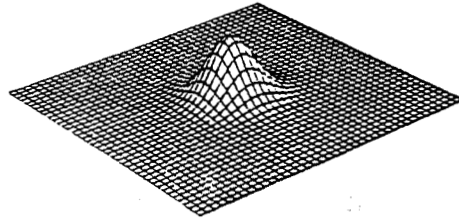
DECISION REGIONS

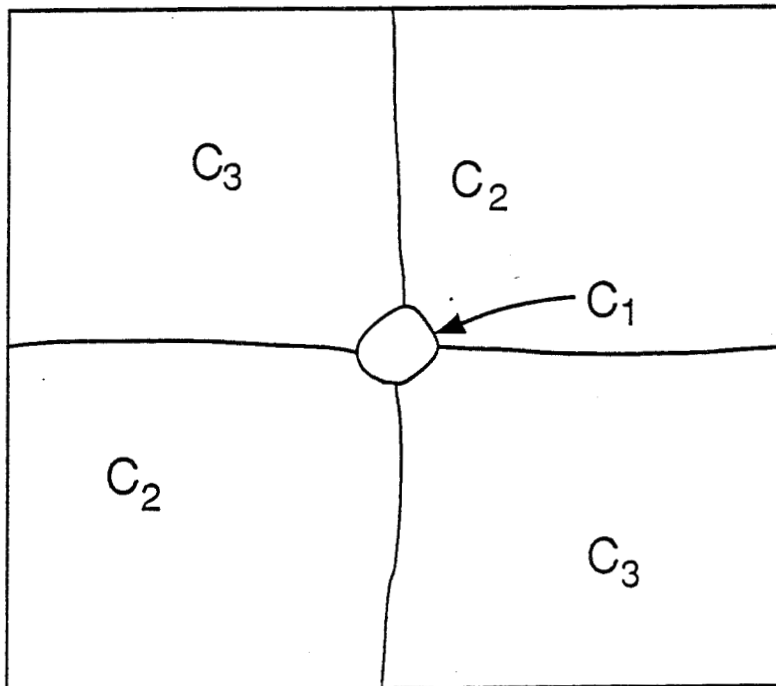


$$0.9 < \sum Y_i < 1.1$$

Radial Basis Function Networks (Gaussian nodes):

- Well-placed classification boundaries
- Check sum $\sum Y_i = 1$ tends to be satisfied *everywhere* (i.e. no regions flagged as novel)

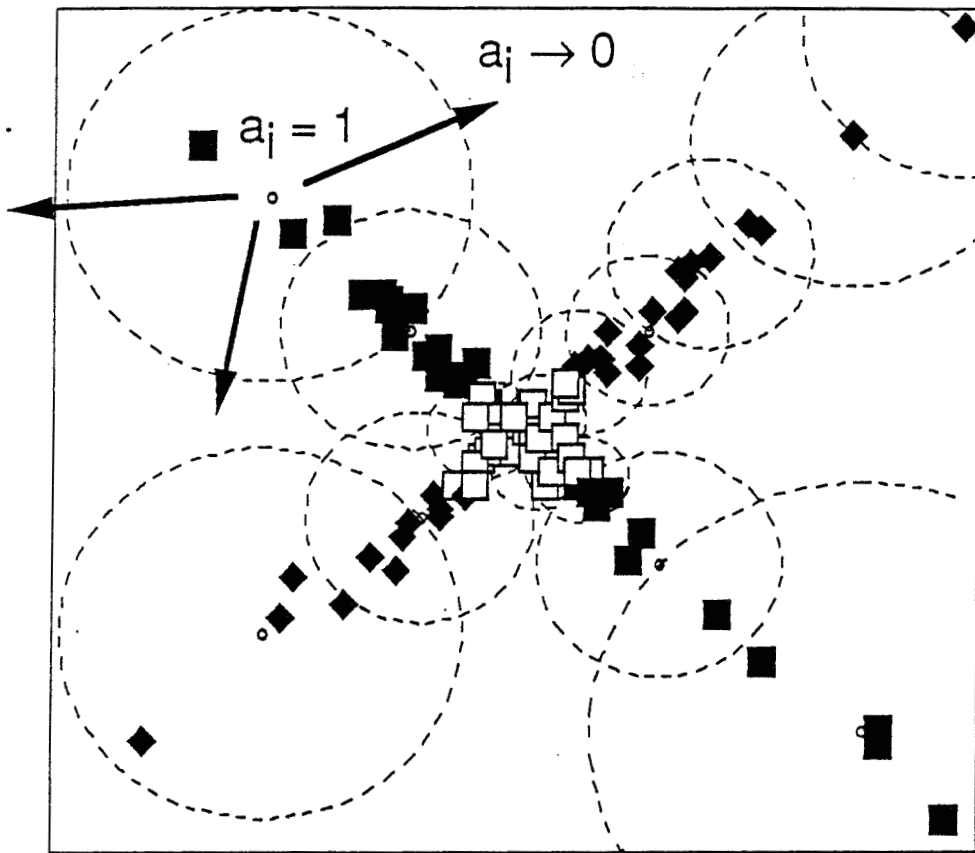




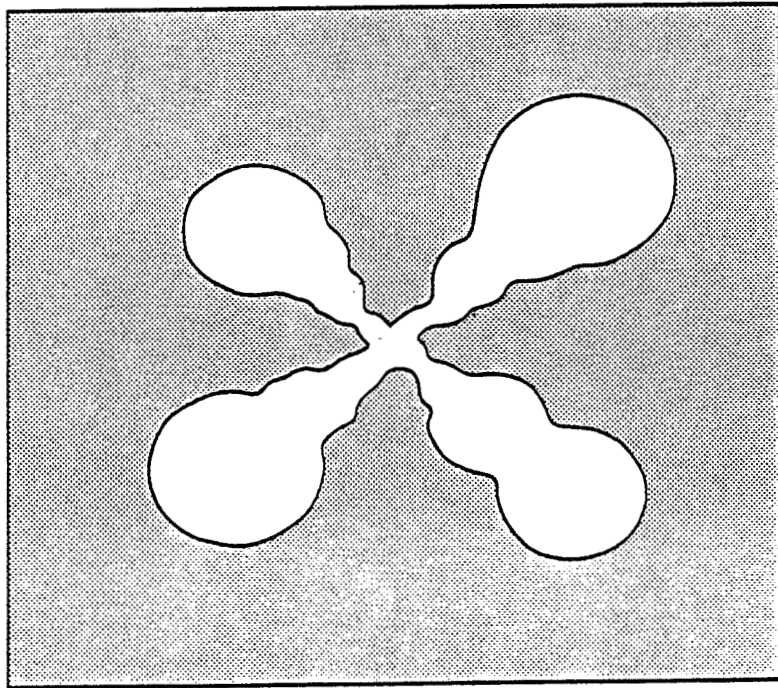
DECISION REGIONS

● Closer look at novelty in RBFNs:

- Radial units are centered among groups of data by k-means clustering.
- Gaussian activation functions $a(x)$ decrease to 0 as one moves away from unit center



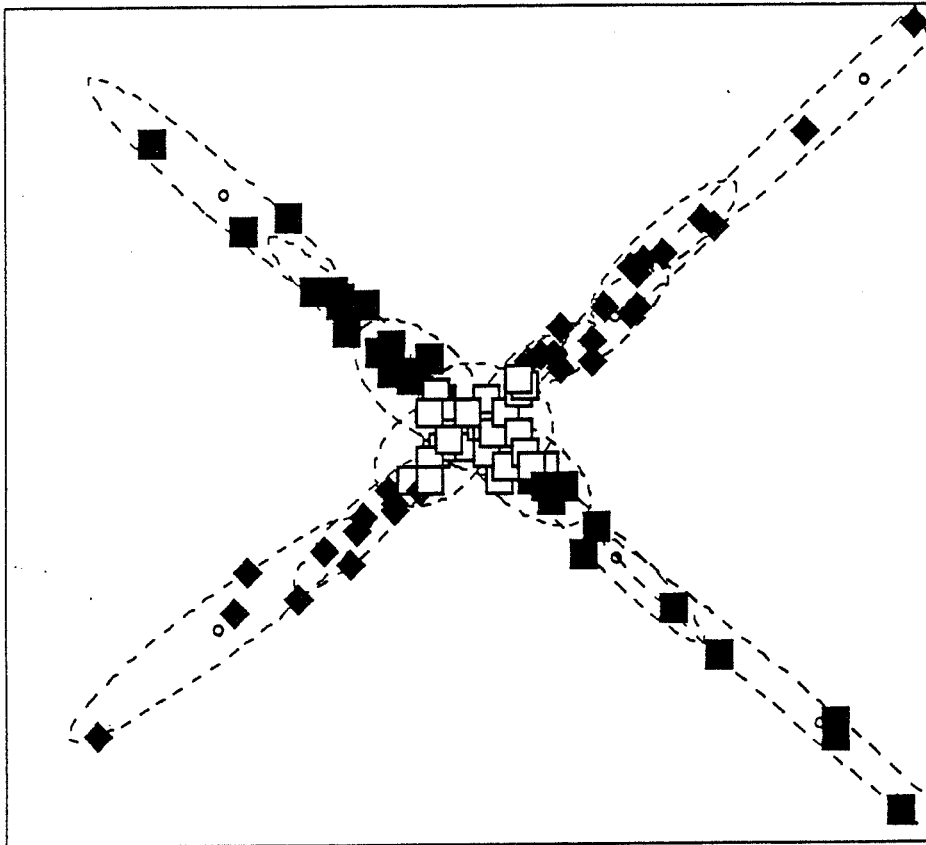
- Novelty can be indicated by $\max(a_j) < \text{cutoff value}$ (e.g. 0.5)
- Works even better with elliptical units



Hidden node
activation > 0.5

Elliptical Basis Function Networks (EBFN)

- Similar to RBFN but unit shapes can be elliptical
- Shapes determined by local covariance structure of data
- Good novelty detection and classification properties



Elliptical Basis Function Coverage of
Fault Classification Data

Data Density Estimation using RBFNs

In each radial or elliptical unit, local data density is approximately:

$$\rho_h = \frac{\text{\# data points local to unit } h}{\text{Volume of unit } h \cdot \text{total \# data points}}$$

A smooth data density estimate at every point in space then given by the interpolation formula:

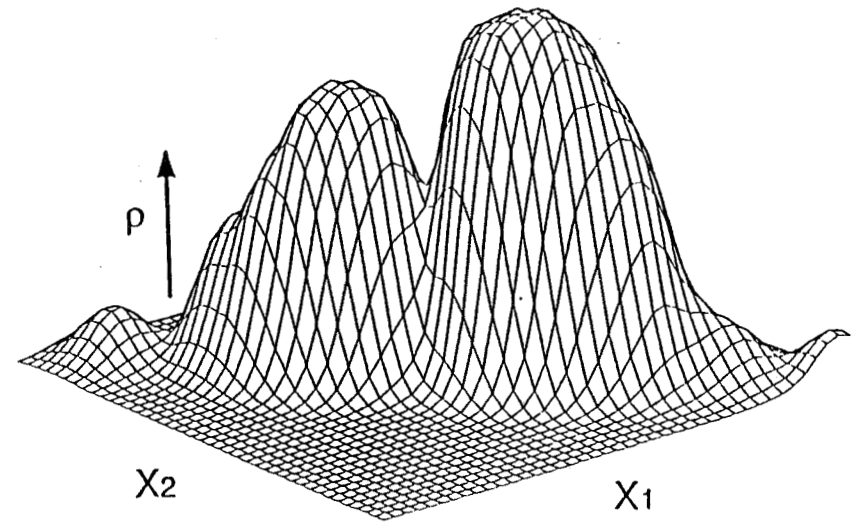
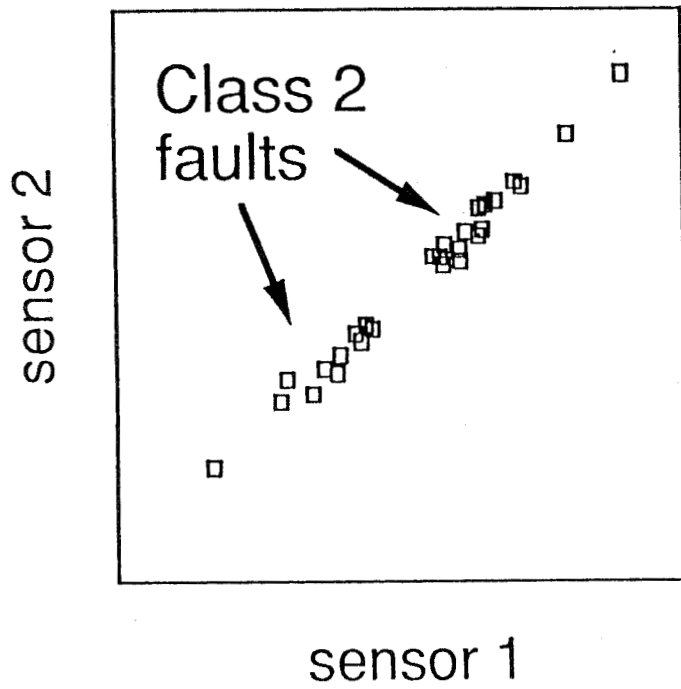
$$\rho(\underline{x}) = \frac{\sum_{h=1}^H a_h(\underline{x}) \rho_h}{1 - \max(a_h) + \sum_{h=1}^H a_h(\underline{x})}$$

Uses of probability density function:

- 1) Class-based decomposition of classifier
- 2) Fault detection using only normal data
- 3) Rectification of sensor faults

(See Leonard, Kramer & Ungar, Comput Chem Eng, vol. 16, 819, 1992)

CLASS-SPECIFIC DENSITY ESTIMATION



Estimated density

Class-Based Network Decomposition

$$p_i(\underline{x}) = P(\underline{x} | H_i)$$

Density related to posterior fault probability conditional on data via Bayes' Theorem:

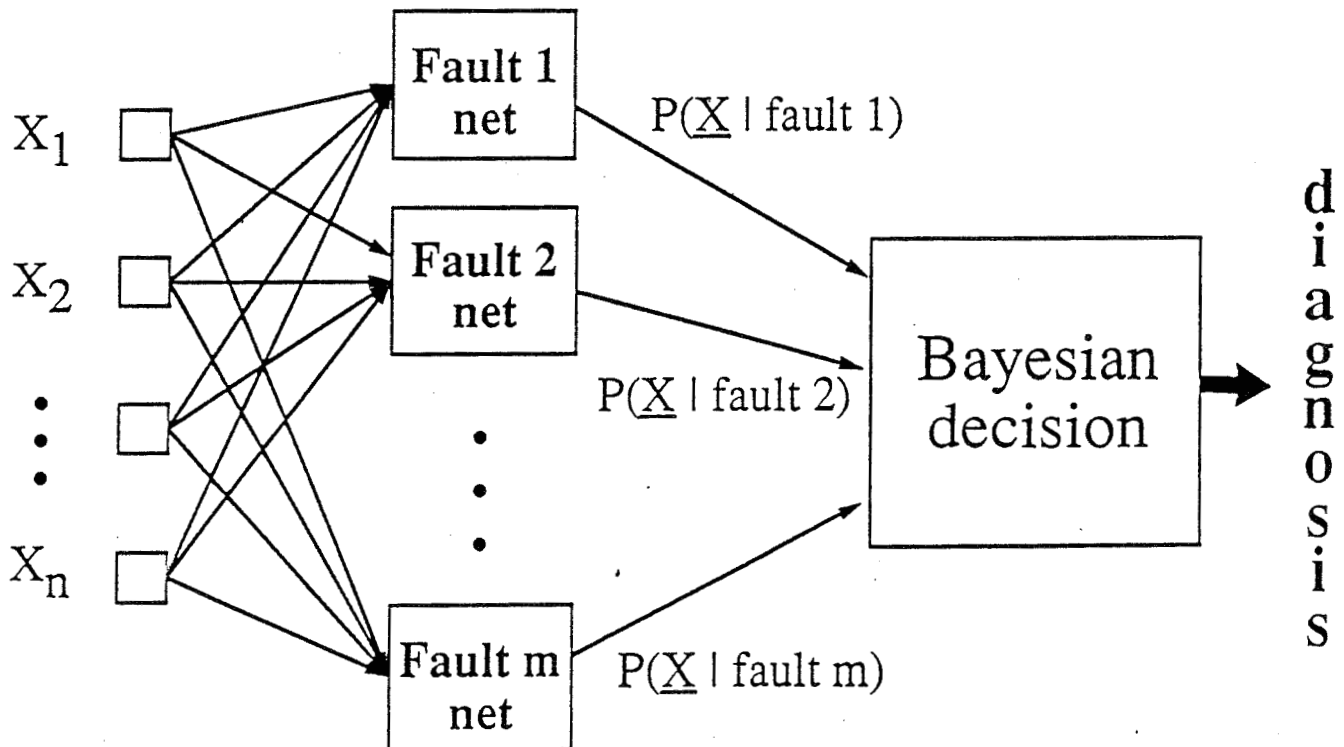
$$P(H_i | \underline{x}) = P(\underline{x} | H_i) P(H_i) / P(\underline{x})$$

$P(\underline{x})$ is pooled density function (all classes).

Relative probability, eliminate $P(\underline{x})$:

$$R_{i|k} = P(\underline{x} | H_i) P(H_i) / P(\underline{x} | H_k) P(H_k)$$

Class-decomposed network:



Decomposition benefits:

- No regression of weights

Work savings = $O(HM/N)$

H = # hidden nodes

M = # faults

N = # inputs

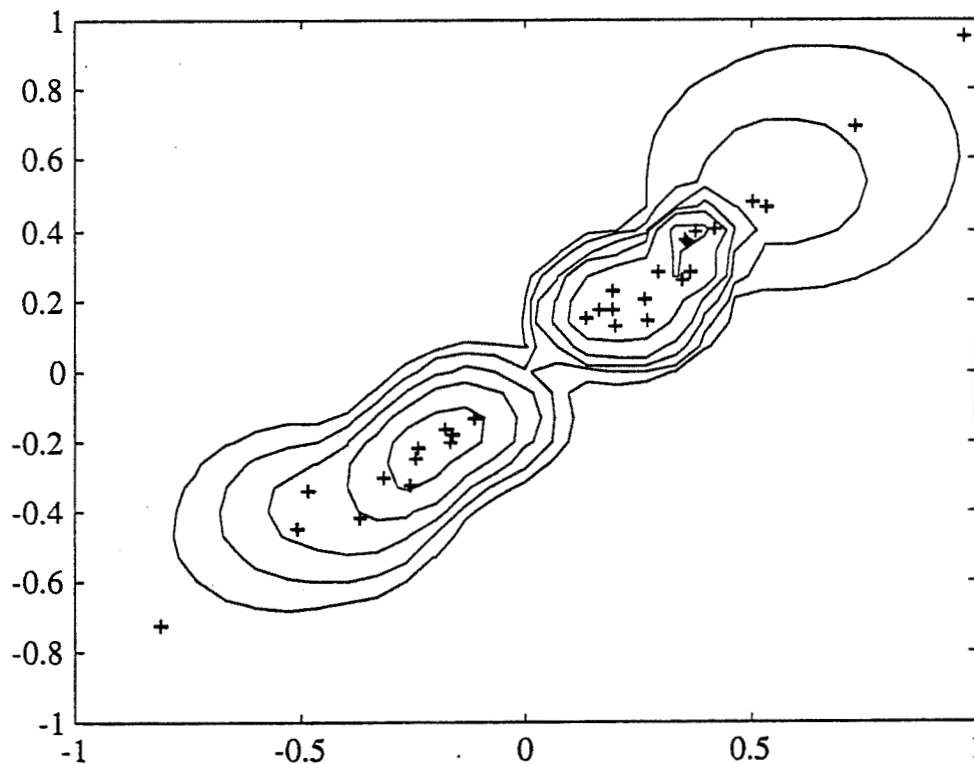
- Allows incremental development of classifier, easy incorporation of new data
- Prior probabilities and misclassification costs can be incorporated in Bayesian decision

FAULT DETECTION USING DENSITY FUNCTION

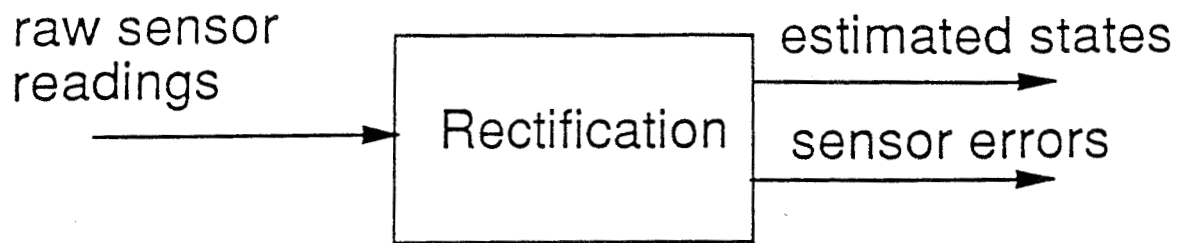
Fault detection: Is current state in the normal class, or out?

Fault data not needed.

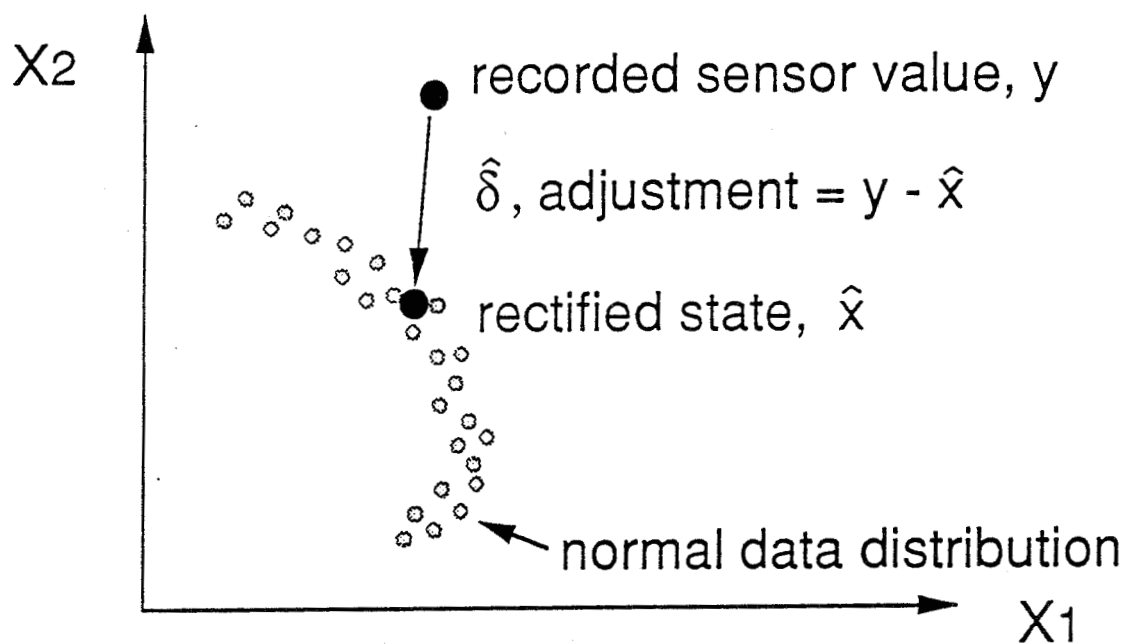
- Model probability density function of normal class
- Place probability limits, e.g. 95% for declaring fault



Rectification of sensor faults by probability optimization



- Assume model of normal probability distribution
- Hypothesize sensor failures only

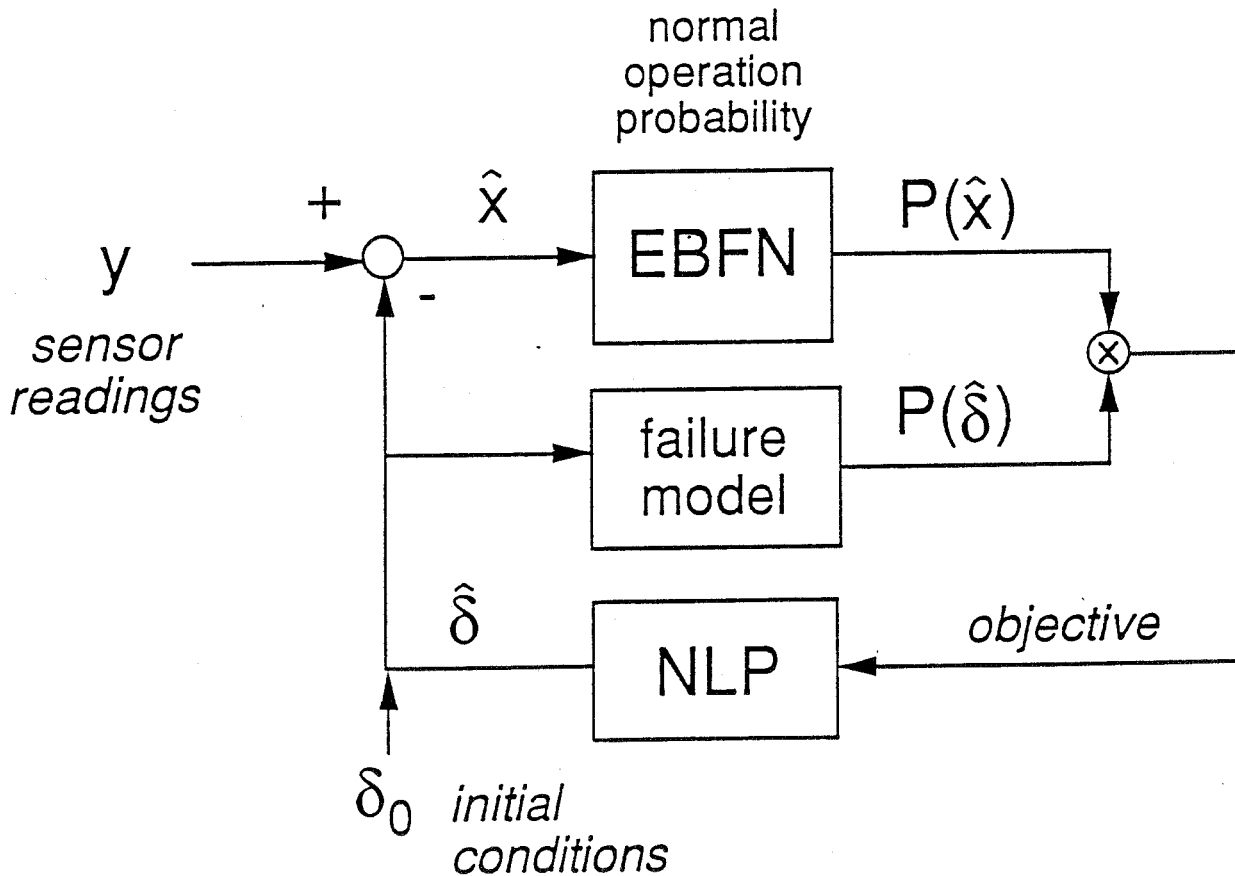
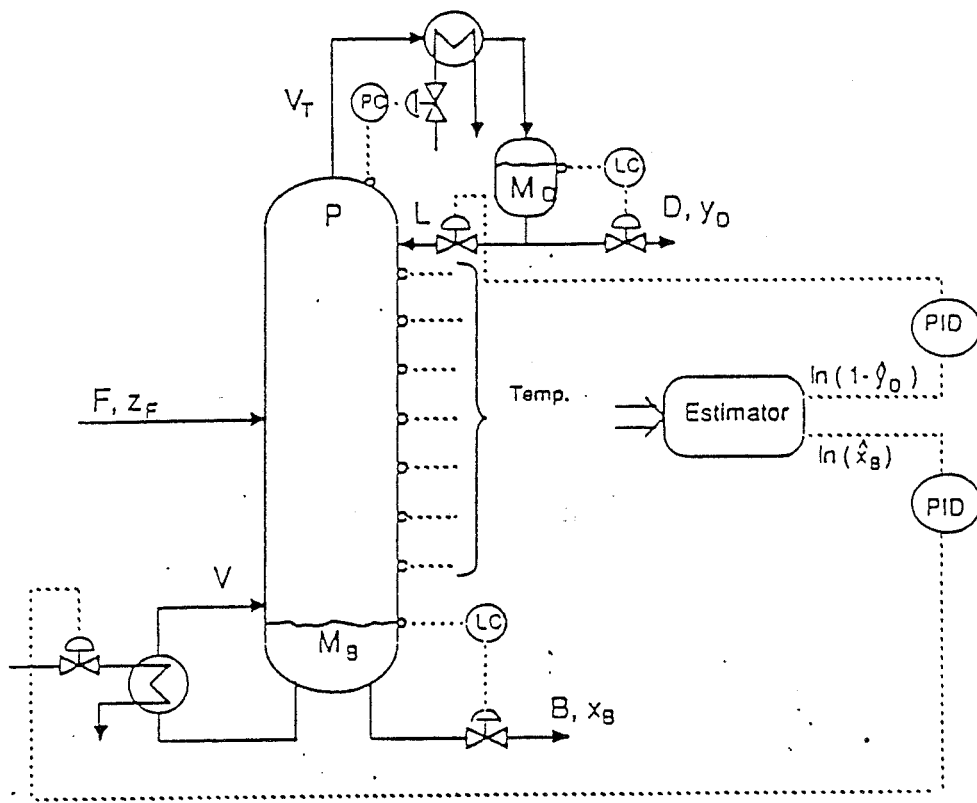


$$\underset{\hat{x}}{\text{maximize}} P(\hat{x}|y) \propto P(\hat{\delta})P(\hat{x})$$

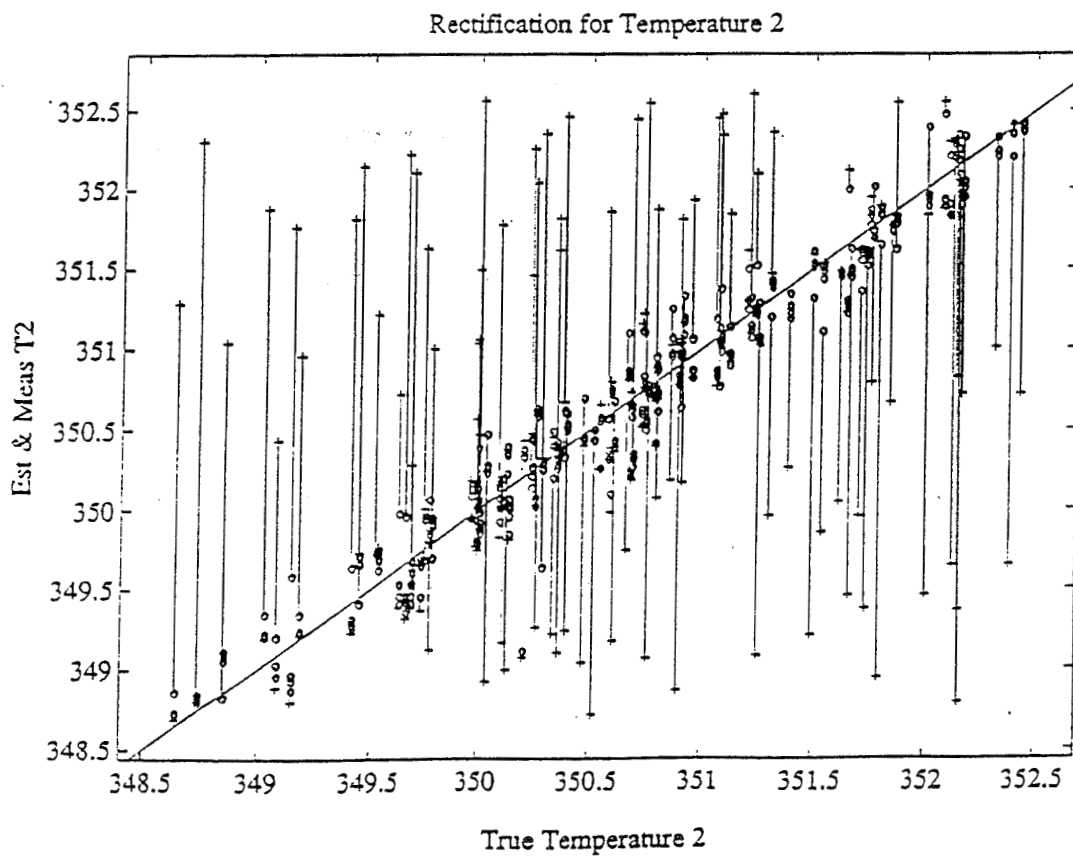
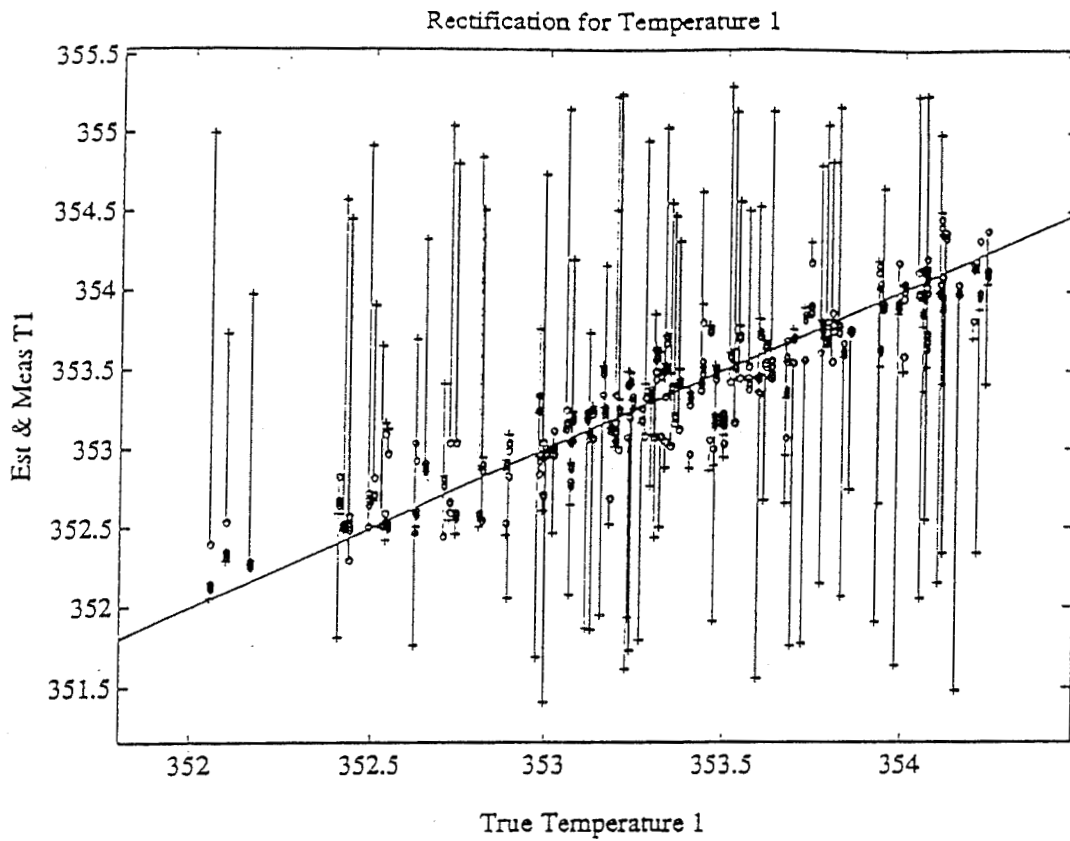
likelihood of
adjustment

likelihood of
rectified state

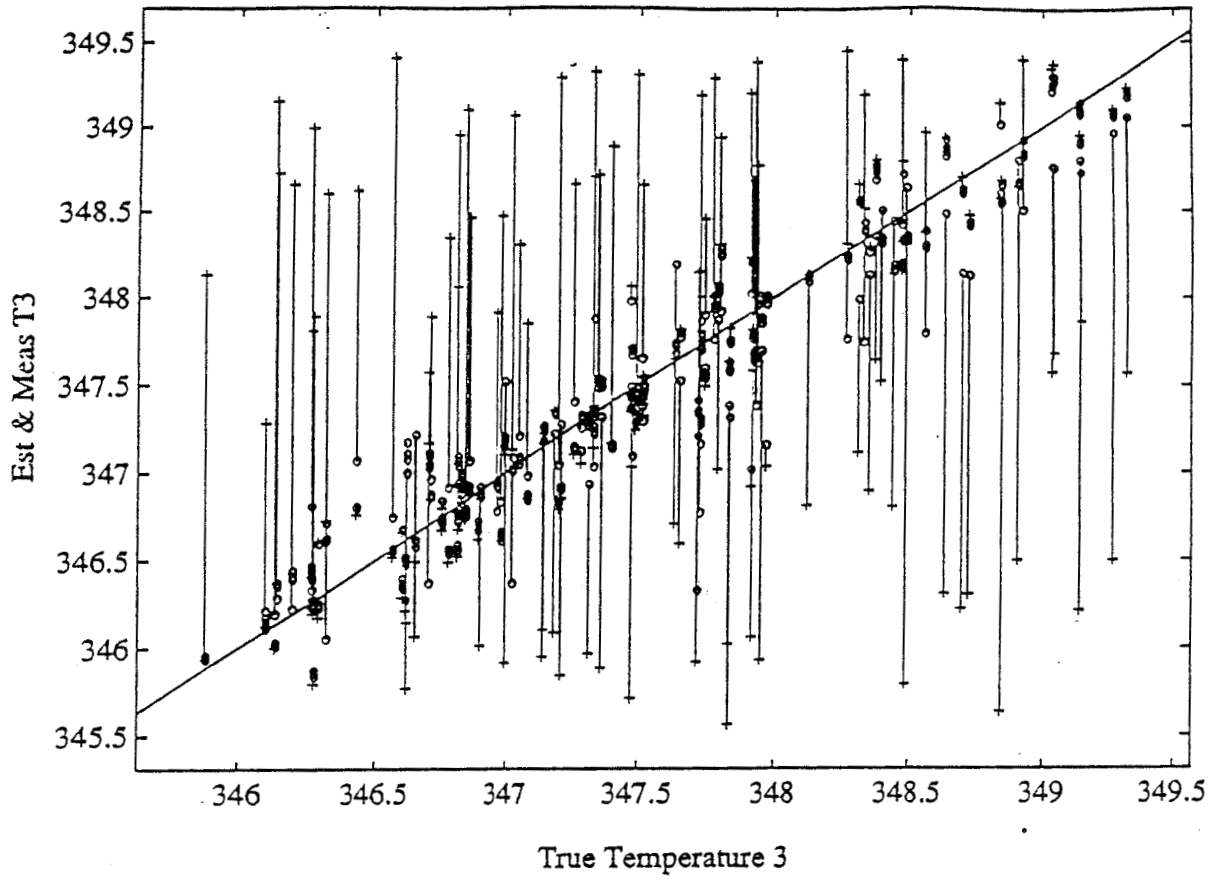
Example: Plate temperature rectification in distillation



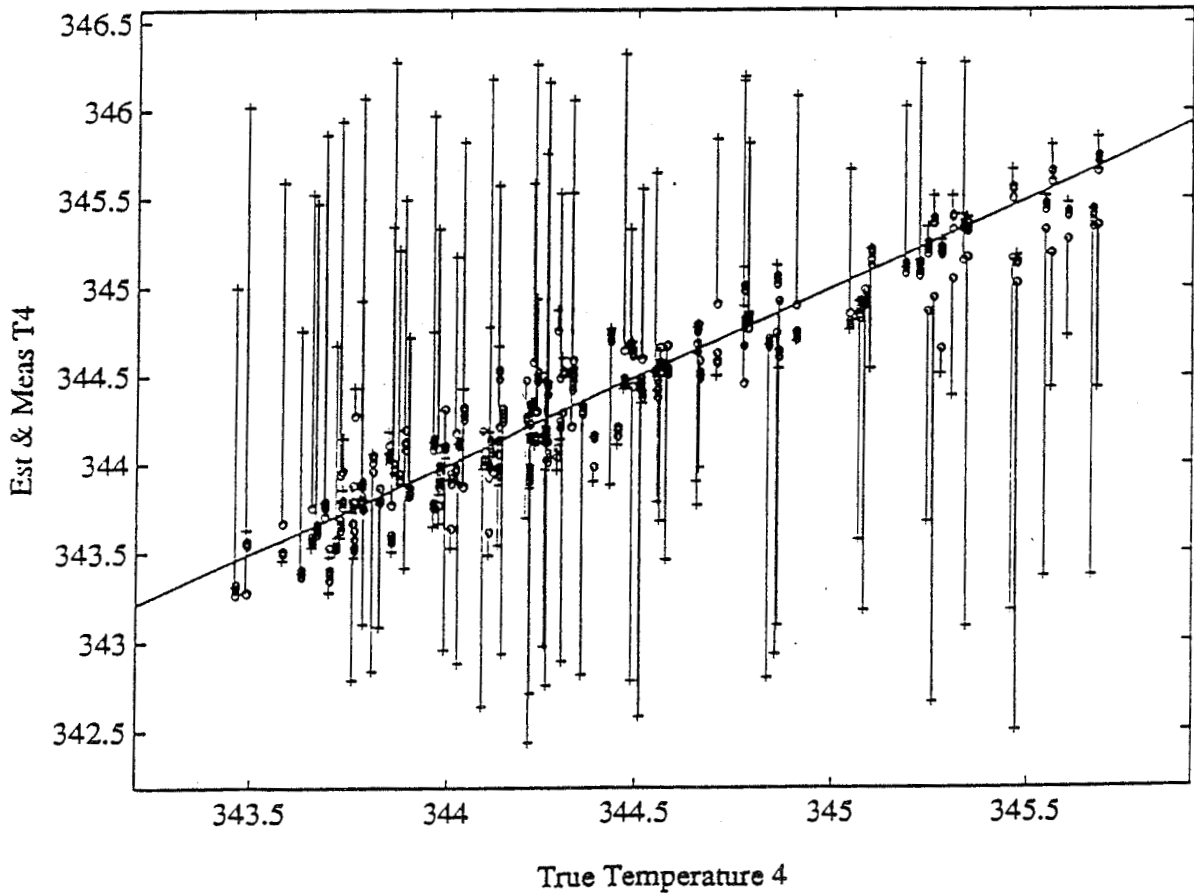
Test: Corrupt each of 5 sensors in test set of 100, yielding 500 examples with single sensor failure.



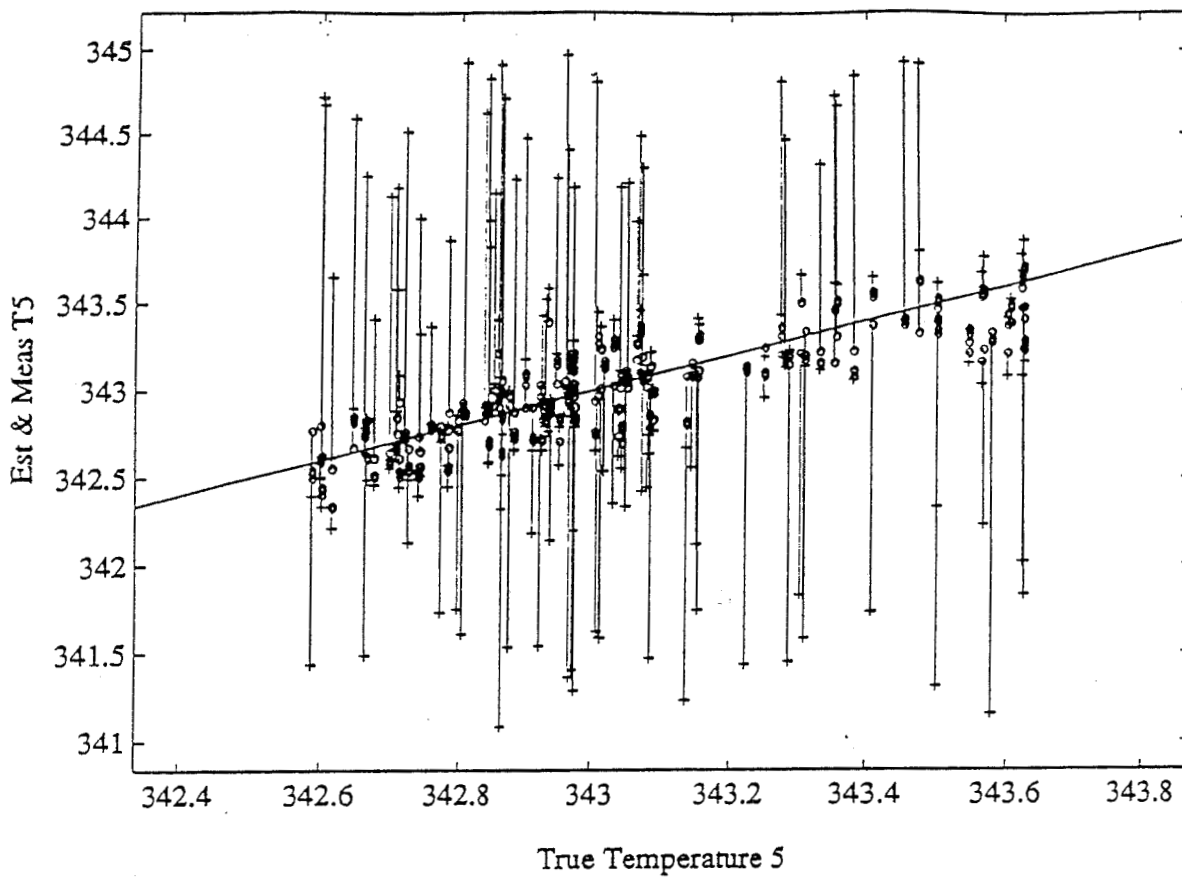
Rectification for Temperature 3



Rectification for Temperature 4



Rectification for Temperature 5



Summary & Conclusions:

Diagnosis can be approached by:

- Multi-class training
- Single-class training

Multi-class diagnosis yields relative fault probabilities

Radial basis function networks preferred approach to multi-class diagnosis

Single-class training involves extraction of statistical distribution model

Can be approach using radial basis functions

Useful for:

- Decomposing multi-class problems
- Fault detection
- Rectification of sensor data

omit

Session VII

FORMAL METHODS

9:00 - 11:00

Session Co-Chairs:

Damir Jamsek, *Odyssey Research Associates*
&
Charles Hardwick, *Univ. of Houston-Clear Lake*

Presenters:

**Report on International Study of Industrial
Experience with Formal Methods: Case Studies & Analysis**

Ted Ralston, *Ralston Research Associates*

Software Development for Safety-Critical Medical Applications
John Knight, *University of Virginia*

Ada Code Verification Using Penelope

Damir Jamsek, *Odyssey Research Associates*

Formal Methods at IBM FSC

David Hamilton, *IBM Federal Systems Co.*

Dr. Charles Hardwick is Professor of Information Systems at the University of Houston-Clear Lake. He came to the University of Houston-Clear Lake in 1981. He is currently serving as Co-Chair of the Information Systems Research Division of RICIS. His research interests are in the area of formal methods, and the application of information technology within a changing work environment.

Theodore J. Ralston

Mr. Ralston is currently President of Ralston Research Associates, a sole proprietorship consulting firm in software engineering and European technology analysis based in Tacoma, Washington. Prior to starting RRA in January 1991, Mr. Ralston was the European Technology Analyst in the Microelectronic and Computer Technology Corporation's (MCC) International Liaison Office. In 1988 while at MCC Mr. Ralston helped launch a new research project in formal methods in MCC's Software Technology Program. This project, which finished in December 1991, was oriented toward transition to industry of formal methods, tools, and processes.

Prior to joining MCC in 1983, Mr. Ralston was on the staff at Stanford University's Center for International Security and Arms Control where his responsibilities included teaching, research, and developing a computer database on international security and arms control. Prior to Stanford, Mr. Ralston served for seven years on the professional staff of the U.S. Senate Select Committee on Intelligence and the staff of Senator Warren G. Magnuson. His duties included intelligence analysis of strategic and tactical weapons intelligence, arms control agreement verification, intelligence community budget authorization, foreign political risk assessment, and legislation and executive orders pertaining to the United States intelligence community.

Mr. Ralston received his undergraduate degree in Pre-Medicine from the University of Washington in 1971 (B.Sc in Zoology and B.A. in History) and his graduate degree in history and languages (Russian) from Oxford University in 1974. In 1988, Mr. Ralston was appointed to the National Academy of Science's Committee on International Developments in Computer Science and Technology. He is the author of "The Verification Challenge: Problems and Promise of Strategic Nuclear Arms Control Verification" and a co-author of "Global Trends in Computer Technology and Their Impact on Export Control", and "Finding Common Ground: U.S. Export Controls in a Changed Global Environment", both published by the NAS.

International Study of Industrial Experience with Formal Methods

Ted Ralston
Ralston Research Associates

RICIS Symposium, October 30, 1992

International Study of Industrial Experience with Formal Methods

**Ted Ralston
Ralston Research Associates**

RICIS Symposium, October 30, 1992

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Outline of Talk

- 1. Purpose of Study and Sponsors**
- 2. What Are Formal Methods?**
- 3. Cases Studied**
- 4. Method of Conducting Study**
- 5. Criteria and Priorities of Information**
- 6. Three Sample Case Studies**
 - SACEM Train Control System
 - Hewlett-Packard Patient Monitoring System
 - INMOS Transputer
- 7. Some Preliminary Observations**

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Purpose of Study

1. to provide an authoritative record on the practical experience to date;
2. to better inform industry and government bodies developing standards and regulations; and
3. to provide pointers to future research and technology transfer needs

Sponsors

1. National Institute of Standards and Technology
2. Naval Research Laboratory
3. Atomic Energy Control Board of Canada

Study Team

Susan Gerhart, Applied Formal Methods
Dan Craigen, ORA Canada Ltd.
Ted Ralston, Ralston Research Associates

Review Committee

John Gannon, UMaryland
Adele Goldberg, ParcPlace
Lorraine Duvall, Duvall Computing
John Marciniak, CTA, Inc.
Morven Gentleman, NRC/Canada

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

What is a Formal Method

- body of techniques for specifying and verifying systems based on discrete mathematics
- hard to classify; better to see as "roles" played in process

formal SPECIFICATION – use of math notation to build models of systems to define them (for requirements or design) and reason about them

formal DEVELOPMENT – use of math notation and logic to guide, express, and verify steps of a design.

formal VERIFICATION – proof of properties, often using a mechanical theorem prover

formal TESTING – testing from a formal specification ???

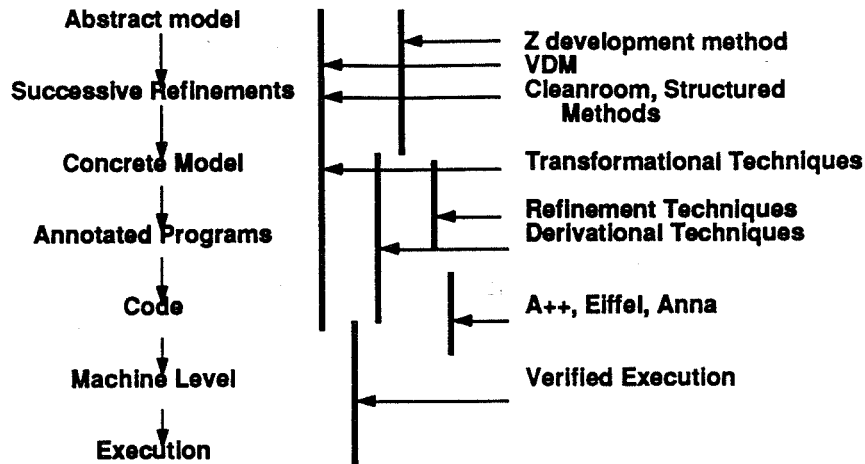
formal FRAMEWORK – use of math notation to build a general set of terms, an architecture, or some other reusable structure covering a set of applications, solutions, etc.

formal DOCUMENTATION – use of math notation to structure user, design, system, etc. documentation, accompanied by other explanatory material

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

How Formal Methods are Used: A Process + Method View



Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

How Formal Methods are Used: A "Role" View

Requirements	<p>Modeling using formal notations to develop a formal specification</p> <ul style="list-style-type: none"> • set based (Z, VDM, HP-SL, Raise) • state-based (STATEMATE, Leveson) • others (box structures in Cleanroom)
Design	<p>Satisfying requirements</p> <ul style="list-style-type: none"> • decompose spec and refine • review, "test", inspect.... • prove if needed
Code	<p>Implementing design</p> <ul style="list-style-type: none"> • derive assertions
Documentation	<p>Communication among designers and users</p> <ul style="list-style-type: none"> • translating math to natural language annotations • transmuting to another representation • maintaining/updating formal spec as system evolves

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Cases Studied

1. CASE/SSADM Toolset by Praxis
2. IBM CICS
3. COBOL Re-structuring/Goddard using Cleanroom
4. Darlington Reactor Control System
5. Large Complex Systems (ESPRIT)
6. Multinet Gateway Network
7. Patient Monitoring System
8. Paris Metro - Sacem Train Control System
9. TBACS Token-Based Access Control System
10. TCAS Traffic Alert and Collision Avoidance System
11. Tektronix Oscilloscope
12. Transputer Specification and Verification

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Issues of Interest in Cases

<u>Case Study</u>	<u>Issue</u>
1. Praxis	FM as "in-house" methods
2. CICS	FM in re-specification/re-engineering
3. Cleanroom	Combining FM, statistical validation, reviews
4. Darlington	FM and reviewability in safety-critical system
5. LaCos (ESPRIT)	Transition of comprehensive FM method/tools
6. Multinet	verification for security using mandated FM
7. HP	technology transfer; use of FM in med. domain
8. Train Control System	top-to-bottom FM process for certification
9. TBACS	small-scale product; non A-1 mech proof
10. TCAS	reviewability of a formal specification
11. Tektronix	FM as a communication tool; FM & reusability
12. Transputer	FM in hardware specification and verification

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

How the Study was Conducted

Data Collection

Literature

2 Questionnaires used to structure interviews

Interviews with managers, developers, and when possible customers

Integration into Q2

Analysis

- Analytic framework:
- evaluation of impact on "product" and "process"
 - vectors (+, 0, -) to indicated impact
 - subjective judgments correlated with interviews and literature
 - critique by Review Committee
 - R&D recommendations

Copyright: Raiston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Evaluation Factors

Product Features

- Client satisfaction
- Cost
- Impact of product
- Quality
- Time-to-Market

Process Features

General Process Features

- Cost
- Impact of Process
- Pedagogical
- Tools

Specific Process Features

- Design
- Reuse
- Maintainability
- Requirements Capture
- V&V

Copyright: Raiston Research Associates, October 1992

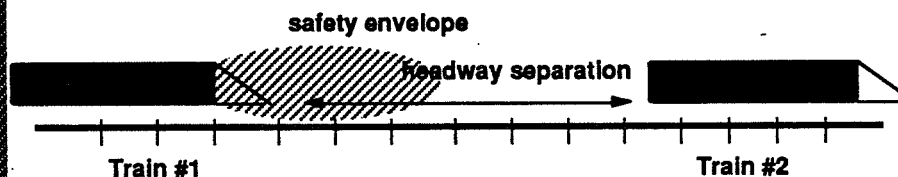
International Study of Industrial Experience with Formal Methods

Sample Case: Paris Metro SACEM Train Control System

Application Domain: Safety-Critical System - Paris Metro, Line A: 20 km/200 trains/60,000 passengers/hour
Customer: RER (Paris Subway Authority)
Developers: GEC-Alsthom, RATP, and CSEE
Problem: use "new digital technology" in safe way
Requirements: (1) Reduce "headway" from 2 min 30 sec. to 2 min even; use computer control ("new technology")
(2) obtain safety certification from RATP
System: SACEM consists of a specially encoded 68000-class microprocessor and software control system, plus analog trackside signalling and switching equipment
Duration: 11 years (1978-89)
Formal Method: Hoare Assertions and Abrial's "B" Method

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods



Challenges:

- several different types of rolling stock to detect, some SACEM equipped and others not
- several different types of trackside beacons, transponders, and signal lights, both analog and digital signals
- specially encoded single processor rather than complex synchronized multiprocessor – single processor to be as failsafe as possible
- getting the train "home" when SACEM fails

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

SACEM Project

The Process:

late 1970s	Explored fault tolerance, discovered proof of correctness techniques, did safety studies
early 1980s	Built prototypes for simulations, verified SC code using Hoare assertions, worked with authorities on safety issues, set up special national safety committee
mid-late 1980s	applied B method to validate previous Hoare proofs, continued simulations, brought SACEM on-line in 1989
1990s	applying full scale "B" method to follow-on projects (SNCF trains, Calcutta Subway, and Korean trains); commercializing tools used

Copyright: Raiston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

SACEM Project

Statistics:	16000 lines of safety-critical code (Modula 2)
	132 procedures proved formally
	153 procedures tested semi-globally
	146 semi-global test cases
	347 global tests (scenarios)
	Total Development Time: 315,000 hours (prototype & final)
	Validation of SC software: 33,000 hours (final system only)

Copyright: Raiston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

SACEM Project

The Results:

Verification demonstrated to be viable addition to simulation by adding more assurance without excess cost

validation

Specification used as medium for by RATP

Data:

Breakdown of validation effort by task

Total hours:	145,000
1. formal proof	32.4%
2. module testing	20.1%
3. functional testing	25.9%
4. B re-spec/verif	21.6%

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

SACEM Project: Evaluation

Product Features

• Client satisfaction	+
• Cost	n/a
• Impact of product	+
• Quality	+
• Time-to-Market	n/a

Process Features

General Process Features

• Cost	0
• Impact of Process	+
• Pedagogical	+
• Tools	+

Specific Process Features

• Design	+
• Reuse	+
• Maintainability	n/a
• Requirements Capture	+
• V&V	+

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Sample Case: HP Patient Monitoring System

Application Domain:	Safety-Critical System - database component of an integrated patient monitoring system for hospital use
Customer:	Hewlett-Packard Medical Products Division
Developers:	HP Bristol Research Laboratory and HP Cardiac Care Systems, Waltham, Mass
Problem:	technology transfer
Requirements:	zero defects while adding new functionality; satisfy perceived FDA process requirements
System:	conventional database component
Duration:	18 months
Formal Method:	HP Specification Language (VDM + histories) embedded in rigorous SWE process

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

HP Patient Monitoring System

The Process:

1985 - 1988/9	Development of HP-SL method, toolset, and education/technology transfer mechanisms by HP Bristol Lab
1988 - 1990	education and training in HP-SL; four precursor projects on different medical devices using HP-SL
1990 - 1992	requirements analysis; formal HP-SL spec; coding; unit and functional testing; acceptance testing

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

HP Patient Monitoring System

The Results: successful completion of deliverable;
 termination of formal methods group at HP

Data: Breakdown of effort (number of days per month per task):*

Month	Spec	Design	Code	Test	Other
May	8	0	1	0	6
June	8	3	0	0	8
July	4	11	2	1	4
Aug	0	6	0	0	4
Sept	4	6	4	2	4
Oct	1	3	7	5	3
Nov	0	2	6	10	3
Dec	1	0	8	4	1
Jan	0	3	6	1	2

Totals		
Stage	# days	%
Spec	26	18
Design	34	23
Code	34	23
Test	23	16
Other	31	21

* source: HP Bristol Labs

Copyright: Raleigh Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

HP Patient Monitoring System: Evaluation

Product Features

- Client satisfaction +
- Cost 0
- Impact of product 0
- Quality +
- Time-to-Market 0

Process Features

General Process Features

- Cost 0
- Impact of Process +
- Pedagogical +
- Tools +

Specific Process Features

- Design +
- Dev. Reusable Components n/a
- Reusing existing components n/a
- Maintainability n/a
- Req'ts Capture +
- V&V +

Copyright: Raleigh Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

INMOS Transputer

Application Domain: 3 Inter-related projects

1. Floating Point Unit in sw
2. Verification of T800 FPU
3. Virtual Channel Processor

Customer: INMOS Semiconductor Division

Requirements: IEEE floating point arithmetic standard; concurrent communication between any two processors by single physical link

System: 32-bit microprocessor with on-chip memory and communication

Duration: 6 years (1986 - present)

Formal Method: Z, Occam

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

INMOS Transputer

Process:

1985 Testing problem arose; D.Good talk at RS

1986

1. used Z to specify IEEE standard formally
2. implemented FPU for T400 in software
3. used Z and Occam transforms to verify T800 FPU hardware (along with simulations)

1987

4. Occam Transformation System tool
5. new design of T9000 begun - no FM
6. complex design problems emerge in T9000 pipeline architecture and VCP

1988

1989

1991

7. FM group brought into development effort
8. pipeline problem solved at Oxford, and partial verification of VCP design completed

1991

1992

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

INMOS Transputer

Results:

1. T400 FPU resulted in 10 pages of Z + English
2. Derived T800 FPU hw spec from T400 spec
3. After 4 years 2 errors discovered in FP microcode (translation error and "typing" error - no errors in FM part)
4. Occam Transformation tool
5. 3 month saving in coverage testing (\$1.5 M saved)
6. T9000 effort led to several results:
 1. method for proving correctness of state machines (claimed to be "complete" ??)
 2. automated tool for refinement checking
 3. FM group made part of critical design plan

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Findings

1. FM maturing steadily but slowly
2. FM are being applied in system development of significant scale and importance to the organizations
3. The primary uses of FM are in ASSURANCE, ANALYSIS, COMMUNICATION, "BEST PRACTICE", and RE-ENGINEERING
4. FM for system CERTIFICATION is emerging.
5. Tool support necessary for industrialization but not necessary nor sufficient for applying FM
6. Technology transfer is occurring but very slow and limited
7. FM skills are gradually building within organizations
8. FM applied to code level only evident in a few instances with many limits and hurdles
9. Inadequate metrics and cost-benefit models

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Patterns

- 1. Duration of technology transfer**
 - 3 years minimum to launch/experiment
 - 6 years to see what really works
 - 9 years to produce significant improvements
- 2. Significance of systems actually developed**
 - not large (much less than 100 KLOC)
 - large consequences (i.e., safety, security, loss of revenue)
 - what is the right measure, if there is one?
- 3. Economics**
 - not as time consuming or costly as myths would have us believe
 - training is manageable

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Patterns

- 4. Expertise close at hand**
 - new ideas need this for transition
 - commitment by experts to organization
- 5. FM plays role as communication media**
 - force closer attention to detail
 - provide common basis
- 6. Mandating vs. "softly, softly" approach**
 - MoD 0055 taken seriously but...
 - opportunistic problem solving

Copyright: Ralston Research Associates, October 1992

International Study of Industrial Experience with Formal Methods

Patterns

7. Technology vs. methodology

- methodology makes early inroads
- technology very weak but used

8. FM as a "carrier"

- complex package of ideas that often comes with big need for SOME change
- other ideas "carried", e.g., reviews

9. Slow to spread

- within an organization slow
- across application lines

Copyright: Ralston Research Associates, October 1992

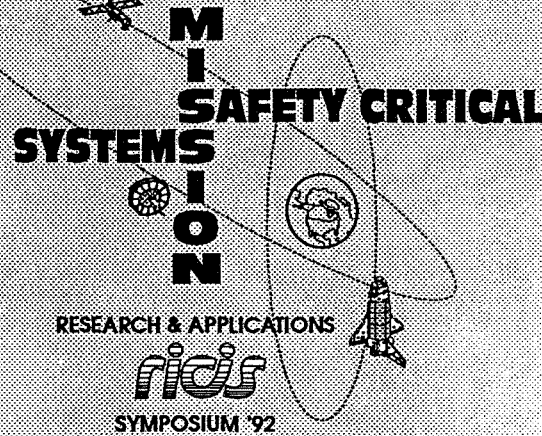
International Study of Industrial Experience with Formal Methods

Technology Transfer Lessons

1. Role of expert or "guru" significant factor in success
2. Involving management at the right point (i.e., when it means something to them)
3. Package FM as part of larger methodology
4. FM brought in at time of major new technology or business opportunity
5. There appear to be 3 stages, possibly 4
6. Developing meaningful metrics for FM will aid transfer

Copyright: Ralston Research Associates, October 1992

N95-14171



75-61

27245

P-14

1995/07/57

324074 16A

SOFTWARE DEVELOPMENT FOR SAFETY-CRITICAL MEDICAL APPLICATIONS

John C. Knight

ABSTRACT

There are many computer-based medical applications in which safety and not reliability is the overriding concern. Reduced, altered, or no functionality of such systems is acceptable as long as no harm is done. A precise, formal definition of what software safety means is essential, however, before any attempt can be made to achieve it. Without this definition, it is not possible to determine whether a specific software entity is safe. A set of definitions pertaining to software safety will be presented and a case study involving an experimental medical device will be described. Some new techniques aimed at improving software safety will also be discussed.

BIOGRAPHY

Dr. Knight is a professor of computer science at the University of Virginia. He joined the university in 1981 after spending seven years at NASA's Langley Research Center. His current research interests are in the development of techniques to support software production for safety-critical computer systems.

SOFTWARE DEVELOPMENT FOR SAFETY-CRITICAL MEDICAL APPLICATIONS

John C. Knight
Department of Computer Science
University of Virginia
Charlottesville
Virginia 22903



OUTLINE

- What?
 - What Is Software Safety Exactly?
 - A Framework Of Definitions
- Why?
 - Why Is Software Safety Important?
 - Who Should Care And Why Should They Care?
- How?
 - How Can Software Safety Be Achieved?
 - What Process, Techniques, And Tools Are Needed?
 - What Questions Remain?
- Case Study:
 - Evaluation Of Proposed Ideas
 - Safety-Critical, Medical Application
- Research Status And Plans



SOFTWARE SAFETY

- Public Exposure To Digital Systems Increasing - Serious Problem
- Several Standards Written Or Being Written, E.g. MoD Def. Std 0055
- Lots Of Papers On Software Safety:
 - Extremely Valuable And Important Contributions To The Topic
 - But, They Tend To Stress System Safety
- Some Important Questions:
 - Precisely When Should Software Be Considered Safe?
 - What Is The Role And Responsibility Of The Software Engineer?
 - What Is “Good Engineering Practice” In This Case?
 - Exactly Who Has Legal Responsibility For What?
- Distinguish Between *Safety* And *Reliability*, And Between *Safety* And *Availability*



WHY IS PRECISION IMPORTANT?

- Concept Is Intuitive And Informal In General - I Know What It Means
- Something Is “Safe” If It Does No Harm - It Had Better Not Harm Me
- Precise Framework Of Definitions Is Important For:
 - Software Engineers
 - Regulatory Agencies
 - Legal System
 - Me
- Software Engineers Need To Know:
 - What Is Required Of Them, Why, And When
 - When Software Is “Good Enough”
- Regulatory Agencies:
 - Responsibility To Protect The Public - FDA, FAA, Etc
- Legal System:
 - Apportioning Blame After An Accident



SOME TIME-HONORED ANECDOTES

- Aircraft Landing Gear Raised While Aircraft On Ground:
 - Test Pilot Input During Ground Test, Aircraft Damaged
 - “Operational Profile or Specification In Error”
- Computer Controlled Chemical Reactor Seriously Damaged:
 - Mechanical Alarm Signal Generated
 - Computer Kept All Controls Fixed - Reactor Overheated
 - “Systems Engineers Had Not Understood What Went On Inside The Computer”
- F18 Missile Clamped To Wing After Engine Ignition:
 - Aircraft Out Of Control
 - “Erroneous Assumption Made About Time For Engine To Develop Full Thrust”
- All Are Important, Very Serious Incidents - Valuable Insight Gained
- What *Exactly* Is The Safety Issue In Each Case?
- What *Exactly* Is The Responsibility Of The Software Engineer In Each Case?

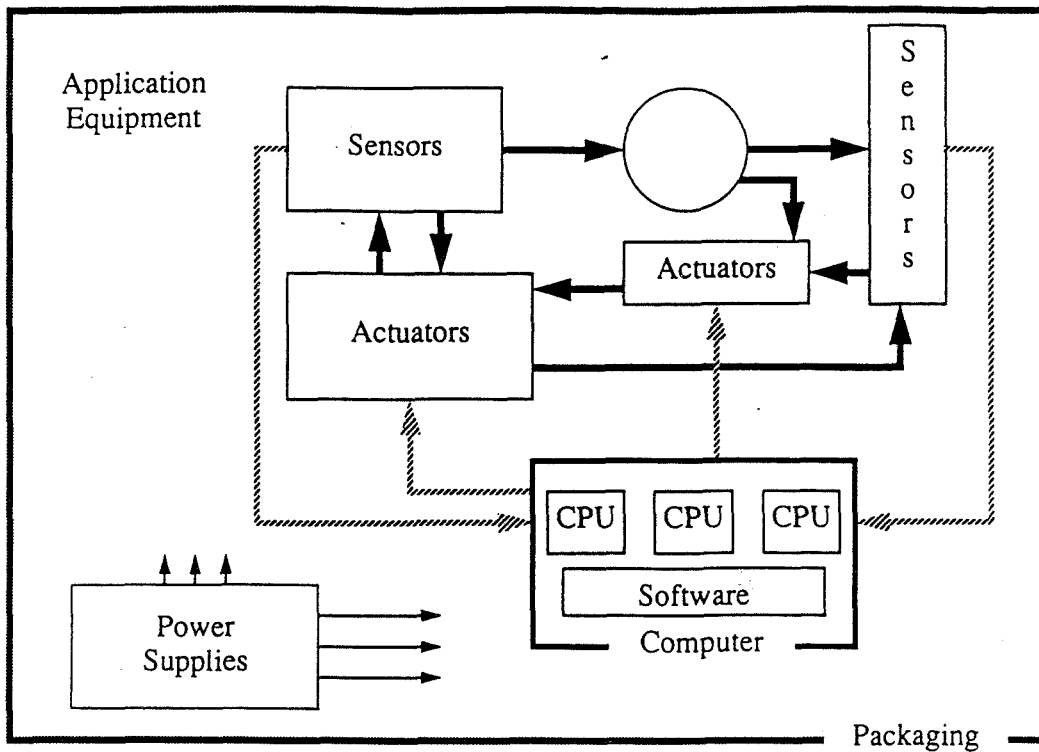


SYSTEM SAFETY

- Informally, *System Safety* Is Subjective
- *Systems Engineers* Have Formalized The Notion Of Safety:
 - Definitions - Hazard, Risk, Acceptable Level Of Risk, *Safety*
 - View System As Well-Defined Collection Of Components
 - Established Practices And Procedures
- Software Researchers And Engineers Trying To Do The Same For Software
- So Far, Success Is Limited
- Within A System:
 - Software Is Merely Part Just Like Computer Hardware, Sensors, Actuators, Etc.
 - Software Can Cause Failure
 - Software Can Prevent Failure
 - Software Can Stand By And Watch Failure Happen
 - But So Can Any Other Part



SOFTWARE SAFETY vs. SYSTEM SAFETY



Medical Software Safety - 7



UVA

Department of Computer Science

SYSTEM CONTEXT FOR SOFTWARE

- Common Observation - In Isolation, Software Is Never Unsafe:
 - True, It Cannot Be Executed In Isolation Either
 - Software Is Useless In Isolation
- Most Components In Any System Are Safe In Isolation
- In Isolation, Software Is Removed From The Notion Of Hazard:

This Does Not Imply That Software Safety Is Meaningful Only In The Context Of The Entire System
- Software Engineers Are Not Qualified To Deal With *Systems* Engineering Issues
- Do We Want The Software Engineer:
 - Deciding Action For Unspecified Input?
 - Implementing Functionality That "Seems Right"?
- Hazards, Risks, Etc. Should Not Appear In The Software Specification
- The *Required Treatment* Of Hazard Must Be Present In The Software Specification

Medical Software Safety - 8



UVA

Department of Computer Science

THE ANECDOTES AGAIN

- Aircraft Landing Gear Raised While Aircraft On Ground:
 - "Operational Profile or Specification In Error"
 - *Systems Engineer's Responsibility*
- Computer Controlled Chemical Reactor Seriously Damaged:
 - "Systems Engineers Had Not Understood What Went On Inside The Computer"
 - *Systems Engineer's Responsibility*
- F18 Missile Clamped To Wing After Engine Ignition:
 - "Erroneous Assumption Made About Time For Engine To Develop Full Thrust"
 - *Probably The Systems Engineer's Responsibility*
- In General, Software Engineer Is Not Trained To Identify Hazards, Consider:
Computerized Flight-Control System Commands Aircraft To Flare On Final Approach At Air Speed Of 128 Knots, Height 180 Feet, 15 Knot Headwind, Throttles At 75%, MLS On, Fuel At 14%, 1,027 Feet From Runway Touchdown Point, Undercarriage Down, Flaps At 30%
- Is This A Hazard?



A FRAMEWORK OF DEFINITIONS

Definition - Component Intrinsic Functionality Specifications

The Required Functionality Of The Component Without Regard To Safety

Definition - Component Failure Interface Specifications

The Required Functionality That Must Be Provided In The Event That The Component Is Unable To Provide Its Intrinsic Functionality

Definition - Component Recovery Functionality Specifications

The Required Functionality That Must Be Provided In The Event That One Or More Other Component Fail

Definition - Component Safety Specifications

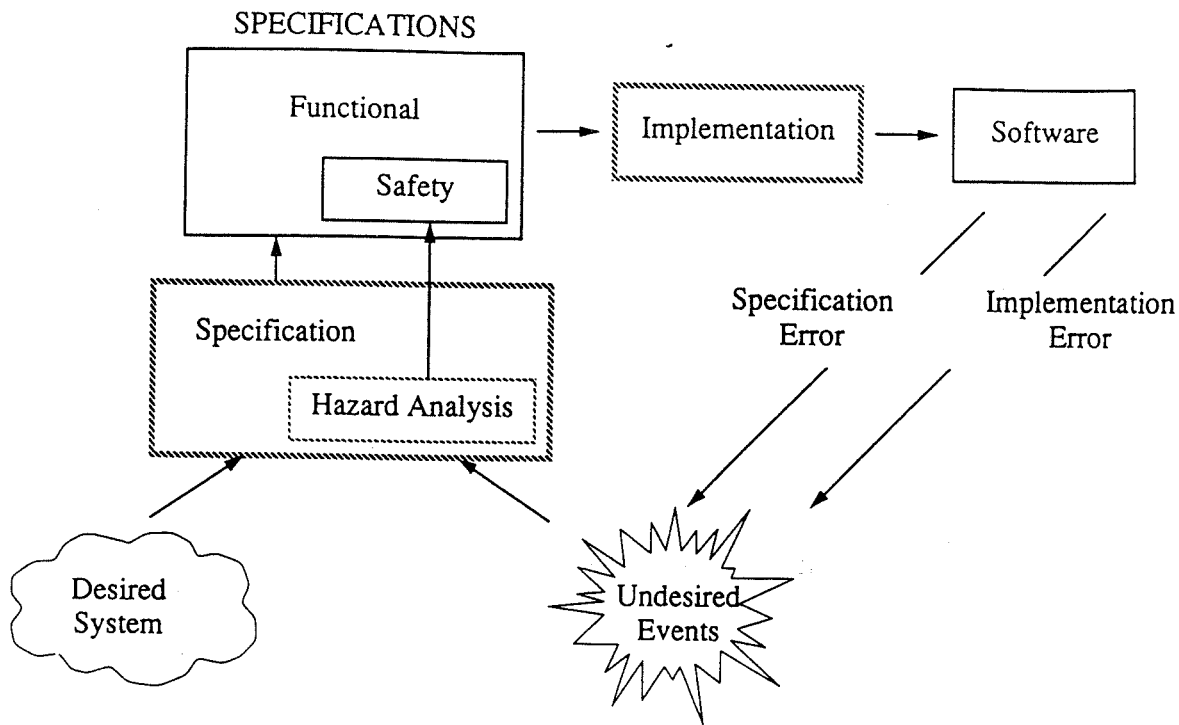
The Component Failure Interface Specifications Combined With The Component Recovery Functionality Specifications

Definition - Software Safety

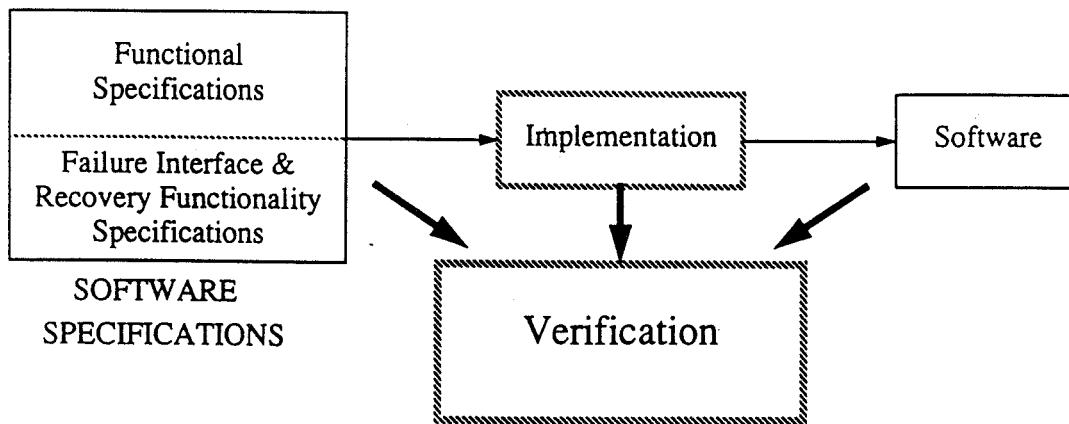
Software Is Safe If It Complies With Its Component Safety Specifications



CONCEPTUAL FRAMEWORK



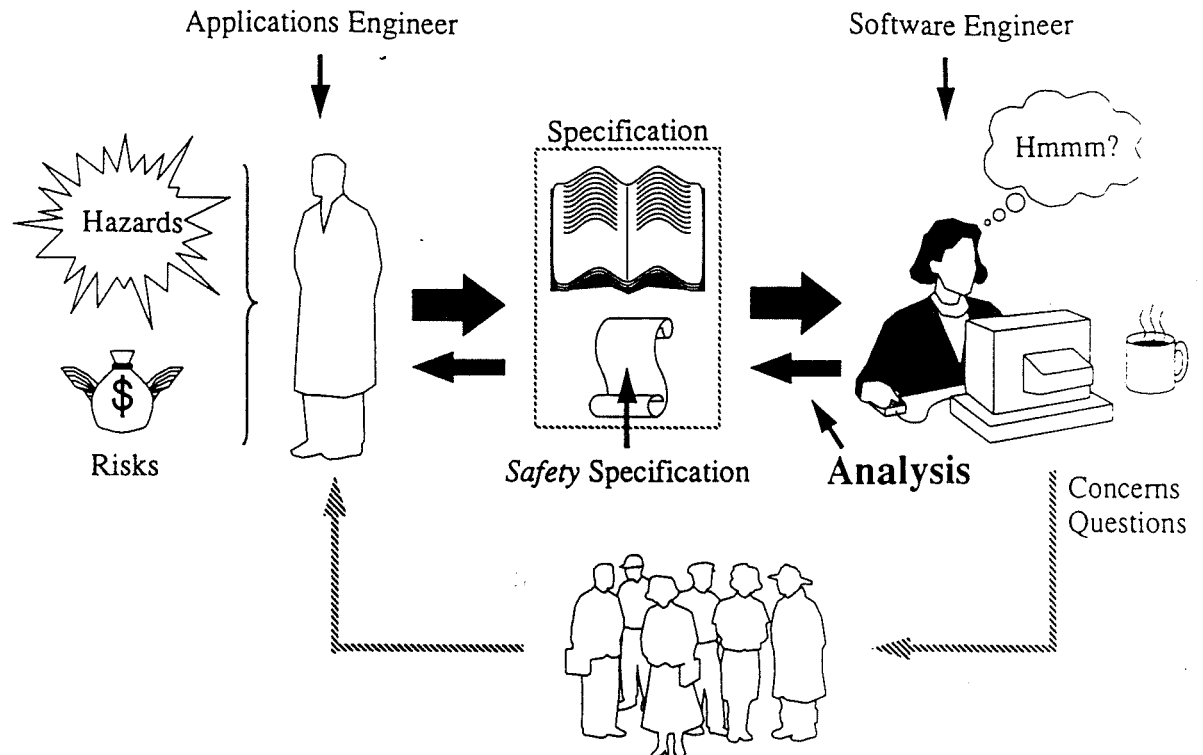
SOFTWARE SAFETY



- Software Is Safe To The Extent That It Complies With Its Safety Specification
- Safe Software Might Fail - That Is A Subjective Issue, Formally It Was Safe
- Software Engineer's Task Now Clear
- Responsibility For Accidents Can Be Fairly Assigned



FORMAL PLACEMENT OF RESPONSIBILITY

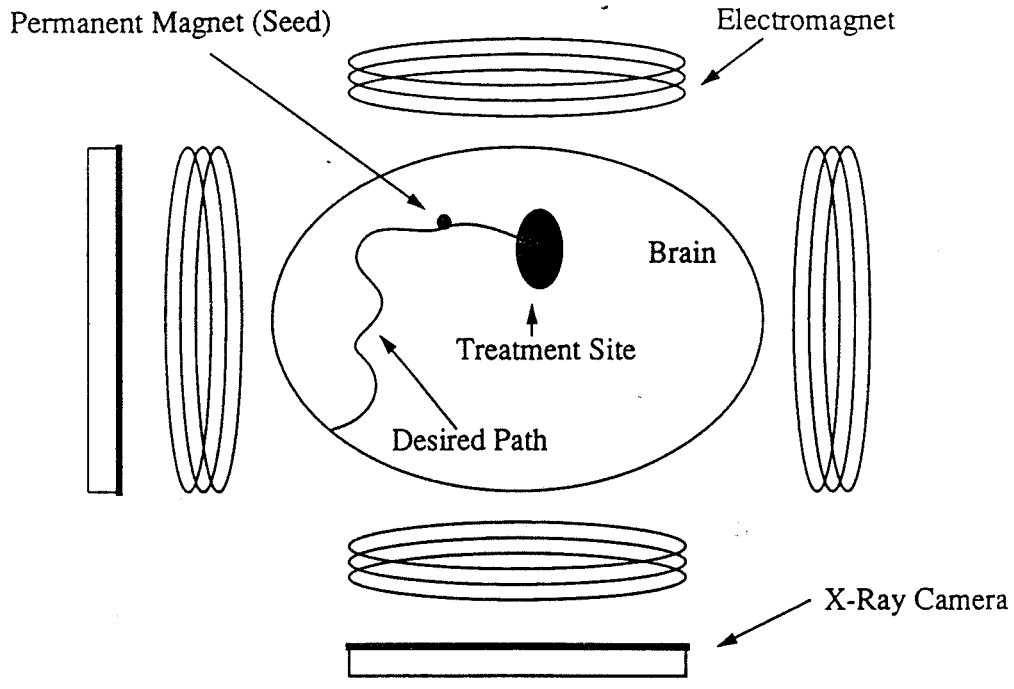


A CASE STUDY

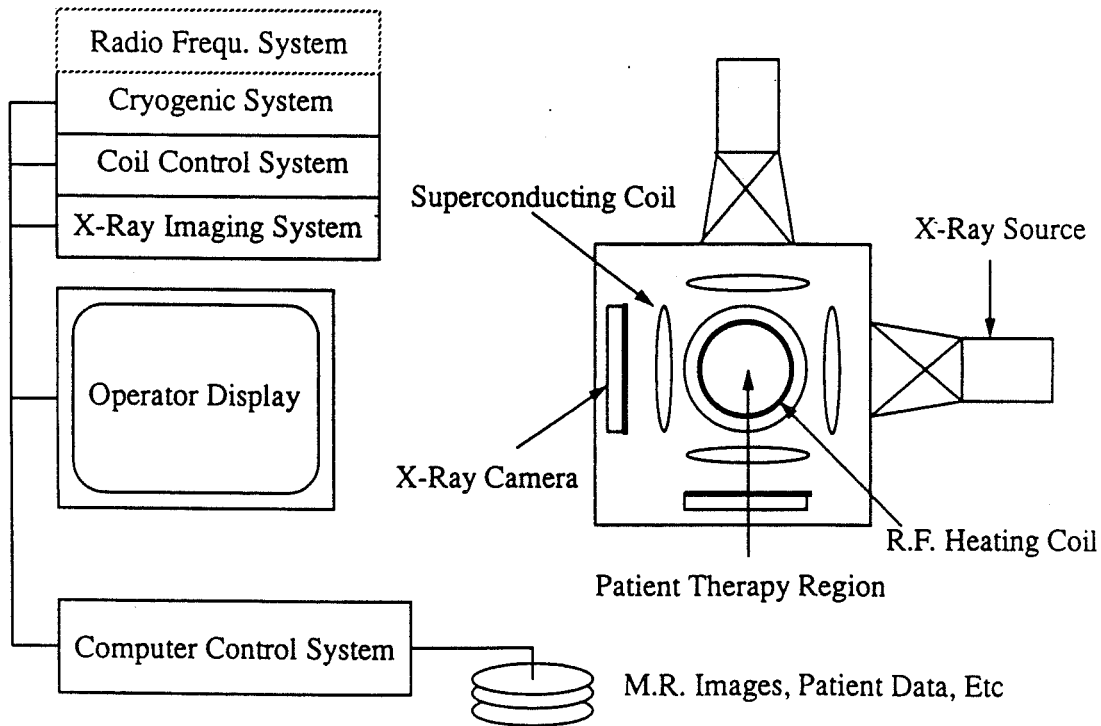
- Is This Conceptual Framework Useful? If Not Why Not?
- If Useful, How Can Safe Software Be Built And Demonstrated?
- Approach:
 - Case Study Based On Safety-Critical Application
 - "Gloves Off", No Assumptions, Not An Academic Study, Do It Right Or Else
- *Magnetic Stereotaxis System (Video Tumour Fighter)*:
 - Experimental Device For Human Neurosurgery
 - Complex Physical System, Clearly Safety-Critical
 - Stringent Safety Requirements
 - Minimal Reliability And Availability Requirements
- Primary Safety Issues:
 - Patient Safety
 - Equipment Safety



MAGNETIC STEREOTAXIS SYSTEM - CONCEPT



MAGNETIC STEREOTAXIS SYSTEM



SOME OF THE MSS/VTF SAFETY ISSUES

- External Superconducting Coils:
 - Incorrect Current Calculated By Software And Applied
 - Coil(s) Fail, Incorrect Coil Shutdown Effected
 - Coil Controller(s) Fail, Incorrect Coil Shutdown Effected
 - Signals Scrambled Between Computer And Coil Controller(s)
- X-Ray Subsystem:
 - Hardware Fails On When Supposed To Be Off Or Vice Versa
 - Software Commands On Incorrectly
 - Image Defects - Ghost, Incorrect, Or "Old" Image Used
 - Incorrect Target Identification - Marker Rather Than Seed
- Radio Frequency System:
 - Hardware Fails On When Supposed To Be Off Or Vice Versa
 - Software Commands On Incorrectly
 - Wrong Power Level Administered



MORE OF THE MSS/VTF SAFETY ISSUES

- Display System:
 - Wrong Seed Location Shown On Magnetic-Resonance Image
 - Wrong Magnetic-Resonance Image Displayed
 - Other Incorrect Data Displayed
- Operator Error:
 - Commands Erroneous Movement
 - Fails To Observe Error Message
- Software System:
 - File System Supplied Erroneous Information
 - Interference From Non-Safety-Critical Elements



DEVELOPING SAFE SOFTWARE

- Framework Of Definitions Is First Step:
 - Now We Know What We Have To Achieve
 - Safe Software Is Well-Defined Target
 - Also Know Who Is *Formally* Responsible For What
- How Is Safe Software Developed?
- There Is No Magic Bullet Is Specified Verification Level Is Very High
- For Safety-Critical Software:
 - No Dependable Technology Exists
 - Many Open Research Areas
 - Safety Is "Simpler" Than Reliability, In Many Cases More Important

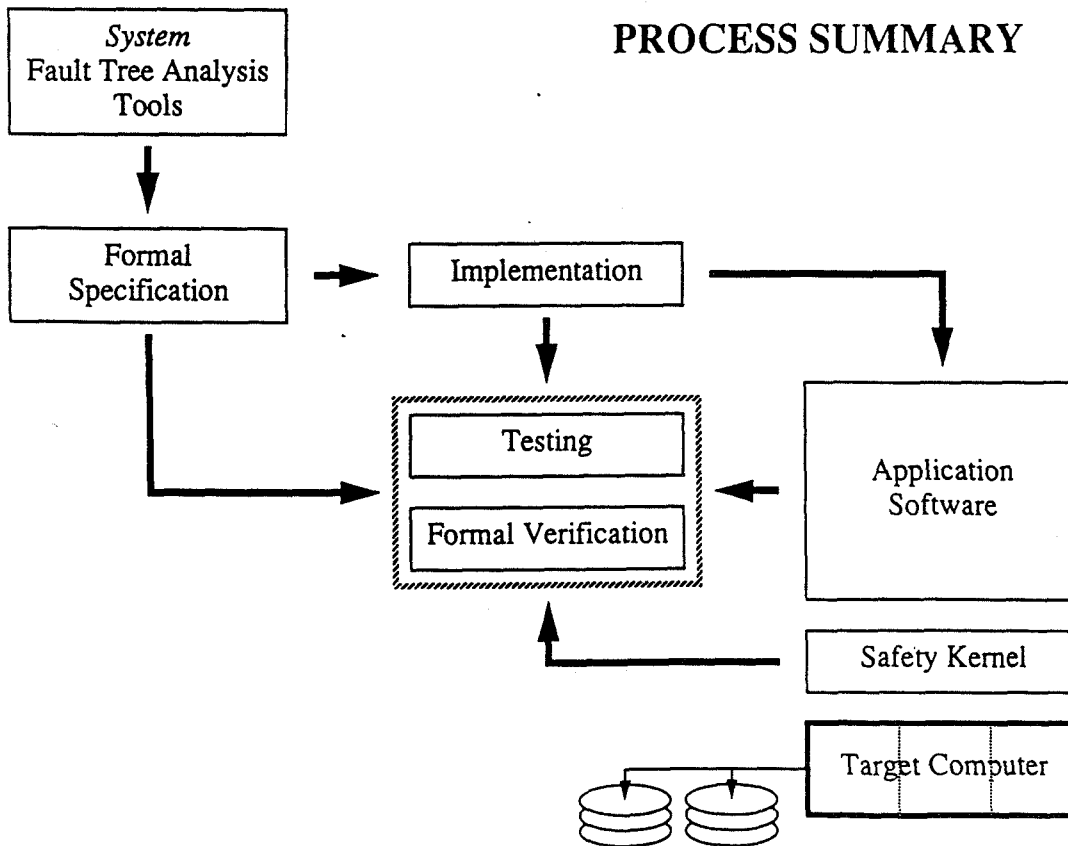


UVA

Department of Computer Science

Medical Software Safety - 19

PROCESS SUMMARY

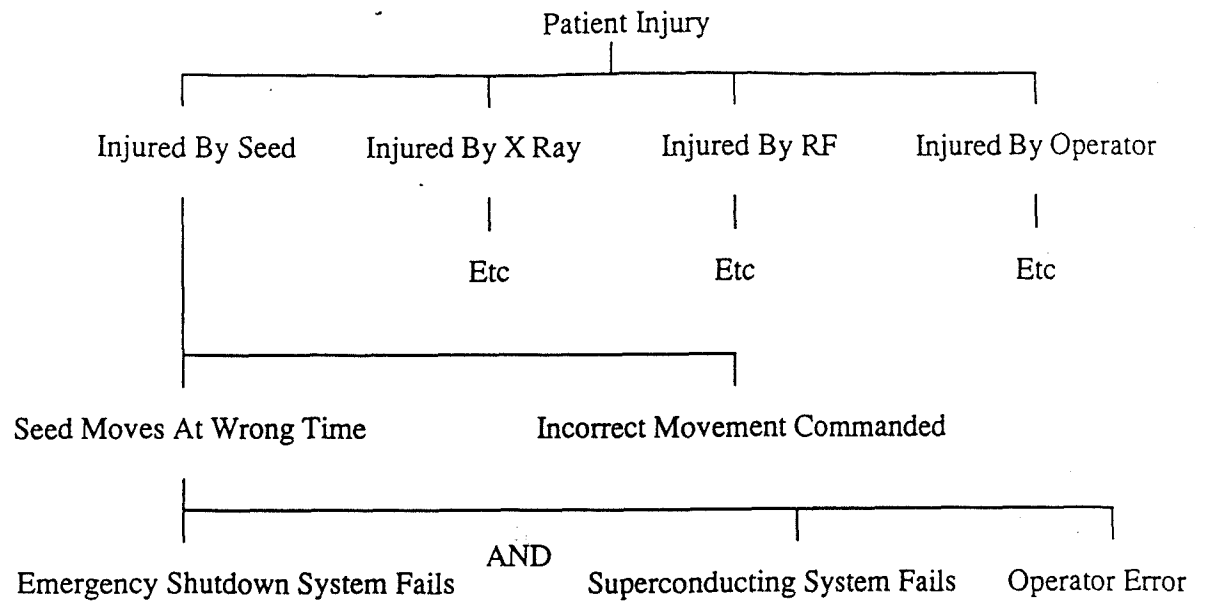


UVA

Department of Computer Science

Medical Software Safety - 20

SYSTEM FAULT TREES



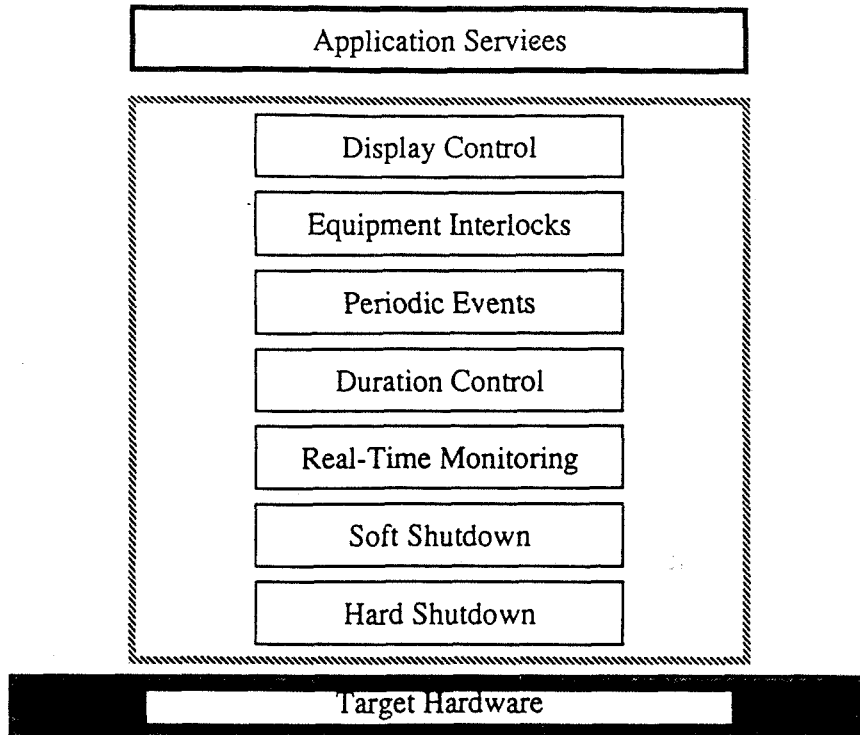
SYSTEM FAULT TREE ANALYSIS

- System Fault Trees Are:
 - Large And Complex
 - Very Hard To Get Right
 - Involve Software In Two Distinct Ways - Software Failure And Software Response
- Tools Needed To:
 - Manipulate Fault Trees To Facilitate Software Analysis
 - Permit Rigorous Software Safety Requirements Process
 - Enable Assurance Of Adequate Coverage By Software
 - Permit Formal Software Safety Specifications To Be Derived
- Tools Concepts:
 - Build Hardware-Only Fault Trees
 - Display, Inspect, Analyze Probabilities, Etc
 - Add Functional Software Nodes
 - Mechanically Derive Software Failure Cases And Required Software Responses
 - Assist With Derivation Of Specifications And Various Property Proofs



SAFETY KERNEL CONCEPTS

- Verifying Safety Properties Is The Single Design Constraint



CASE STUDY EXPERIMENTAL APPROACH

- Develop System Fault Tree And Software Specifications (Drafts Completed)
 - Specifications Presently Written In 'Z' (Draft Completed)
 - Safety Specification Delimited
- Implement Complete, Non-Safe Prototype System Based On UNIX And X
- Add Facilities And Transition To:
 - Safety Kernel On Bare Hardware
 - Progressively "Safer" System
- Verify Safety Properties:
 - Exhaustive Testing - Carefully Avoiding Butler & Finelli's Result
 - Formal Proofs Where Possible
 - Rigorous Argument, Static Analysis, Inspection
- Goal - Repeatable, Dependable Process Providing Assured Software Safety
- Also, A Process That Has Been Evaluated



SUMMARY

- “Invasive” Computer-Controlled Medical Devices Becoming More Common
- Serious Safety Requirements, Often Very Limited Reliability And Availability Needed
- Technology To Deliver Software Safety Is Elusive
- Formalization Of The Meaning Of Terms And The Role Of The Software Engineer
- Software Engineer Is *Not Qualified* For Anything But Software Engineering
- Case Study Being Undertaken To:
 - Evaluate Definitions, Process Concepts, Tools, Techniques
 - Demonstrate Workable Process With Realistic Example
 - Support The MSS Project



Ada Verification using Penelope

**Damir Jamsek
ORA
30 October 1992**

Goal of formal program verification:

- To show implementations conform to specifications
- To make explicit the hypotheses of the system

Penelope depends on:

- Formal language semantics:
 - unambiguous reference and implementation guide
 - provides precise model for analysis
- Formal specification:
 - unambiguous documentation
 - supports and compels thoroughness
- Formal verification:
 - hypotheses made explicit
 - aid to systematic development

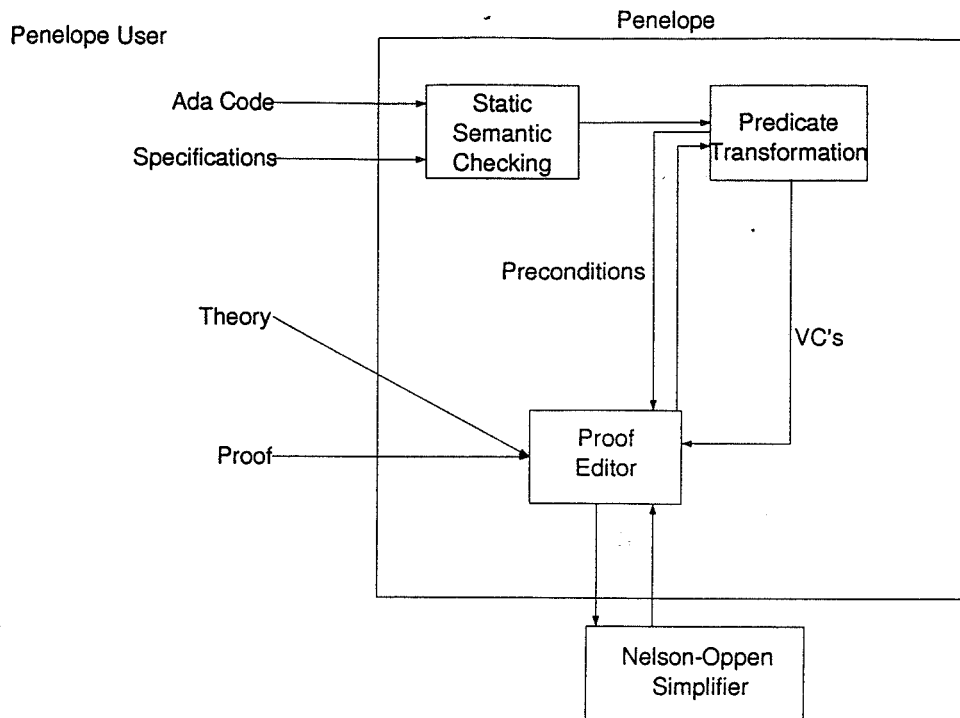
Penelope

Interactive tool for specification and verification of Ada programs

Penelope

- an environment for systematic development of Ada programs
- uses Larch/Ada annotations for specifications
- incremental VC-generation
- incremental simplification and proof of VC's
- mathematical basis

Penelope



© ORA Corp, 1992
SL-0050

6

ORA

Formal basis:

- modeling Ada
- VC generation: predicate transformation
- specification language: Larch/Ada

© ORA Corp, 1992
SL-0050

7

ORA

Embedded assertion:

boolean state function associated with a control point

Asserts the boolean is true whenever control reaches the control point.

Predicate transformers: a form of symbolic execution

-- What must be true here ...

$x := x + 1;$

-- ... if " $x > 0$ " is to be true here?

Answer: $x + 1 > 0$, which equals

$x > 0$ with " $x + 1$ " substituted for " x "

Larch/Ada: two-tiered specifications

Mathematical component - an "environment" of definitions

Interface component - specifies programs in terms of environment

Interface component links "computational" and "mathematical" worlds

- Executable operations $+$, $>$, $=$, ...
- Mathematical operations $+$, $>$, $=$, ...

Mathematical domains of sets, sequences, functions, ...

Connecting the computational and mathematical worlds:

Types are *based on* sorts.

Example: Type `integer` is based on sort `Int`.

```
+: Int, Int → Int  
<: Int, Int → Bool
```

Mathematical components described in Larch Shared Language:

```
GreatestCD: trait  
imports Integer  
introduces  
    gcd: Int, Int → Int  
asserts  
... axioms omitted ...  
implies  
    for all [x, y : Int]  
        reflex: x > 0 → gcd (x, x) = x  
        reduce_left: gcd ((x - y), y) = gcd (x, y)  
        reduce_right: gcd (x, (y - x)) = gcd (x, y)
```

A related interface component:

```
--| USE GreatestCD
FUNCTION gcd (m, n : IN integer) RETURN integer
--| WHERE
--|   IN (m > 0 ^ n > 0);
--|   RETURN gcd(m,n);
--| END WHERE;
```

A *partial correctness* specification

```
function value_of      (the_date  : in string;
                        the_time  : in string;
                        date_form : in date_format;
                        time_form  : in time_format)

return time;
--| WHERE
--|   RETURN strings_to_time (  the_date,
--|                             the_time,
--|                             date_form,
--|                             time_form,
--|                             twentieth
--|                             );
--| RAISE lexical_error <=>
--|   (NOT well_formed_date_string (the_date, date_form))
--| OR
--|   (NOT well_formed_time_string (the_time, time-form));
--| END WHERE;
```

Hypotheses on Ada execution

- No storage or numeric overflow
- Optimizations have not changed the semantics

Hypotheses on Penelope verification

- Specification consistency (No support for verifying the consistency of the mathematical model)

Penelope Theory

- Covers all sequential Ada (less machine dependent areas)
 - address clauses
 - representation clauses
 - machine code insertion
 - unchecked programming
- includes static semantic restrictions against
 - aliasing
 - incorrect order dependence
- preliminary model of concurrency
- consideration of Ada9x issues

Penelope Implementation currently includes:

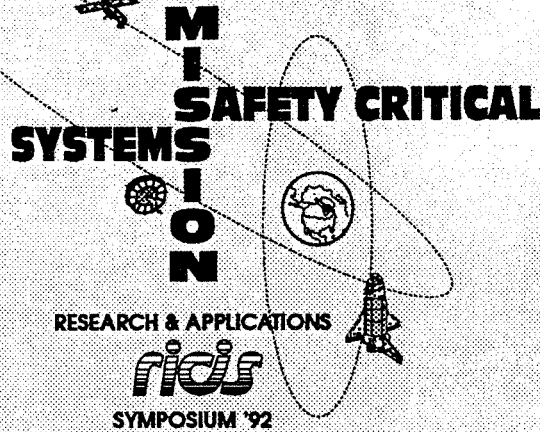
- all sequential statements (except for case)
- packages and private types
- subprograms
- user-defined exceptions and `program_error`
- subset of the type system

under development:

- generics
- remaining type system (including reasoning about constraint errors)

future:

- concurrency



27246

P. 5

1995 107758

327016

51

FORMAL METHODS AT IBM FSC

David Hamilton

ABSTRACT

IBM has a long history in the application of formal methods to software development and verification. There have been many successes in the development of methods, tools, and training to support formal methods. And formal methods have been very successful on several projects. However, the use of formal methods has not been as widespread as hoped. This presentation summarizes several approaches that have been taken to encourage more widespread use of formal methods, and discusses the results so far. The basic problem is one of technology transfer, which is a very difficult problem. It is even more difficult for formal methods. General problems of technology transfer, especially the transfer of formal methods technology, are also discussed. Finally, some prospects for the future are mentioned.

BIOGRAPHY

David Hamilton is an advisory programmer in the IBM Federal Systems Company. He joined IBM in 1982 in a Space Shuttle Flight Software Testing Group and worked in several areas of software testing and test tool development. In 1987, he began investigating tools that supported formal verification, and began studying the reasons why formal methods have not gained more widespread use. More recently, he has been involved in the development of a formal specification language for expert systems and supporting a NASA feasibility study of formal methods. His general interests are software verification and validation, and the use of AI in software engineering.

Formal Methods Technology Transfer

Some Lessons Learned

David Hamilton
IBM Federal Sector Corporation

RICIS Symposium '92

Oct/92

Contents

Introduction and Purpose	1
Harlan Mills and SEW	2
Cleanroom	4
SEDL	6
Stepwise Verification	7
CICS	9
TOP (Verification of ESs)	10
Other Projects and Approaches	11
Note on Quality Emphasis	12
Summary	13
Conclusions	14

Oct/92

i

Introduction and Purpose

- To cover
 1. Some IBM involvement in Formal Methods (FM) projects
 2. A perspective on difficulties of technology transfer (beyond a single project)

- Purpose is not to
 - sell the "IBM approach"
 - argue against feasibility of FM

- Purpose is to
 - learn from other FM technology transfer projects
 - suggest some possible future directions

Oct/92

1

Harlan Mills and SEW

- Mills led massive software engineering education program
 - Software Engineering Workshop was cornerstone
 - 2 week course
 - Taught to all programmers
 - Required to pass final exam

- SEW centered on mathematically-based verification
 - Functional instead of axiomatic
 - model oriented instead of property oriented
 - designed to scale up (stepwise refinement)
 - easier for programmers to understand
 - 2 pieces
 1. Deriving program functions
 - Trace tables (basically manual symbolic execution)
 - Recursion instead of loop invariants
 2. Module-oriented
 - abstract data types
 - constraints/closure on state data (abstract state machine)

Oct/92

2

Harlan Mills and SEW ...

- SEW designed to be practical
 - relatively informal
 - scaled up via abstraction/refinement
 - lots of examples and exercises
 - final test : pass/fail
- Advocated for all programming, not just critical parts
- no support beyond education
 - no tools
 - no consulting
- General reaction was that it was impractical
 - too tedious
 - seemed only for toy problems
- Did not gain widespread use

92

3

Cleanroom

- Named after silicon chip manufacturing environment
- Built on SEW foundation, adding
 - Continuous inspections (SEW style verification)
 - Statistical testing (MTTF prediction)
- Advertised through case studies, not classes
 - Demonstration projects using highly skilled developers
 - To demonstrate benefits
 - To show it can be done, it is practical
- Demonstrations projects were success stories

Oct/92

4

Cleanroom ...

- Showcase project was COBOL/SF
 - Transforms unstructured COBOL into structured COBOL
 - 52,000 SLOCS developed using Cleanroom
 - Results
 - 740 SLOCS / labor month
 - 3.4 errors / KSLOC (before first execution) (70 avg incl. UT)
 - no error ever found during operational use
- Advocacy of Cleanroom continues
 - Widespread use not yet attained
 - But there is a lot of interest in Cleanroom

Oct/92

5

SEDL

- Intended to support SEW/Cleanroom verification concepts
- Built as an extension to Ada
- SEDL compiler generates Ada
- Supports design execution
 - though SEDL generated code may be inefficient
- Includes
 - Abstract data types (set, list, map, etc.)
 - User defined data models
 - model vs. representation
 - constraints
 - Supports mathematical notation
 - $\{X \text{ in CHARACTER : } x \neq 'Q'\}$
 - exists X in S : P(X) and exists Y in T : P(Y)
 - $P > 1$ and not (exists Q in $2..P-1 : P \text{ rem } Q = 0$)
- Use of SEDL is not widespread

Oct/92

6

Stepwise Verification

- Goal: increase use of more formal verification
 - Build on SEW, Cleanroom foundation
 - Investigate tools that support SEW
 - Help projects get started
- Step 1: Understand why SEW approach not widely used
 - Survey of developers
 - Interviewed to those who participated in Cleanroom pilots
 - Results
 - Generally inconclusive, no primary reason(s) found
 - Some themes were:
 1. Lack of experience
 2. Lack of support
 3. SEW reputation

92

7

Stepwise Verification ...

- Step 2: Contact specific projects
 - Demo simple editing tools (support specs)
 - Demo on actual code from the project
 - Discuss methodology
 - Motivate use
 - Offer follow-up consulting
- Results
 - Very positive results on one tool (SEED)
 - Negative results on the methodology
 - redundant work
 - incompatible with current methodology
 - impractical

Oct/92

8

CICS

- CICS is '60s vintage IBM product (assembler) transferred to Hursley, U.K.
- Hursley began big program towards more formal S/W Eng.
- Key approach was formal specs using Z
- Worked hand-in-hand with PRG at Oxford
- Began with selected modules and later expanded use
- Results
 - Some initial "culture shock" for both parties
 - Now some 50 people work directly with Z specs
 - Very positive qualitative results (people like it)
 - Limited quantitative data indicates
 - earlier error removal
 - fewer inserted errors
 - slight cost reduction (9%)
- Use of Z continues at Hursley, but very few other places

Oct/92

9

TOP (Verification of ESs)

- Some concern existed about verifiability of Expert Systems (ESs)
- Study of the problem pointed to one area: Specification
 - Poor low level languages
 - Almost no design
- Led to development of an ES design language
 - Based on work done at USC ISI (LOOM and CLASP)
 - higher level language (term subsumption + OOP)
 - we added annotations
 - Pre and post conditions
 - Global constraints
 - etc.
 - Supported by a compiler (a la SEDL)
 - Supports modularity (a la Ada)
 - Supports annotations
- Cleanroom has also been extended to expert systems
- Neither approach has gained widespread use

Oct/92

10

Other Projects and Approaches

Application above the code level

- Development of a "Box Structures" design language
- Development of a "Box Structures" approach to requirements
- Results
 - SA/SD approach to design most popular new approach
 - Requirements still written in English

● Emphasis on SEW concepts

- Concepts generally well accepted
- Loss of rigor reduces mathematical basis

92

11

Note on Quality Emphasis

● Software quality has extreme emphasis

- Great emphasis on process improvement
- Serious attention given to quality goals and measurement

- Quality motivation programs

- awards and recognition

- Manned Flight Awareness program

● There is willingness to work hard and invest for quality

● The question is not what or how much but how

- FM is generally perceived as not helping

Oct/92

12

Summary

● Goal was to increase the use of formal mathematical approaches to software development (beyond a single project)

1. First through education
2. Then through demonstration projects
3. Then through tool support
4. Then by making methods more practical
5. Finally through direct support (consulting)

● There have been successes

- not nearly as widespread as desired

● This story is not unique to FM

- The problem is with technology transfer, not with technology

Oct/92

13

Conclusions

● Conclusion: Technology Transfer is very hard, even with

- extensive education
- tools support
- demonstrated successes

● Possible future directions

- More consulting ("hand holding") (product champions)
- Use only a core group (FM may just not be for everybody)
- Require use of FM (selected groups)
- Success story close to home
 - technology transfer diminishes rapidly as a function of distance
 - long term commitment is required (success story requires continued follow-up)
- Different formal method(s)
- Different tools (e.g., theorem prover)

Oct/92

14