



110038

The Role of Computers in Research and Development at Langley Research Center

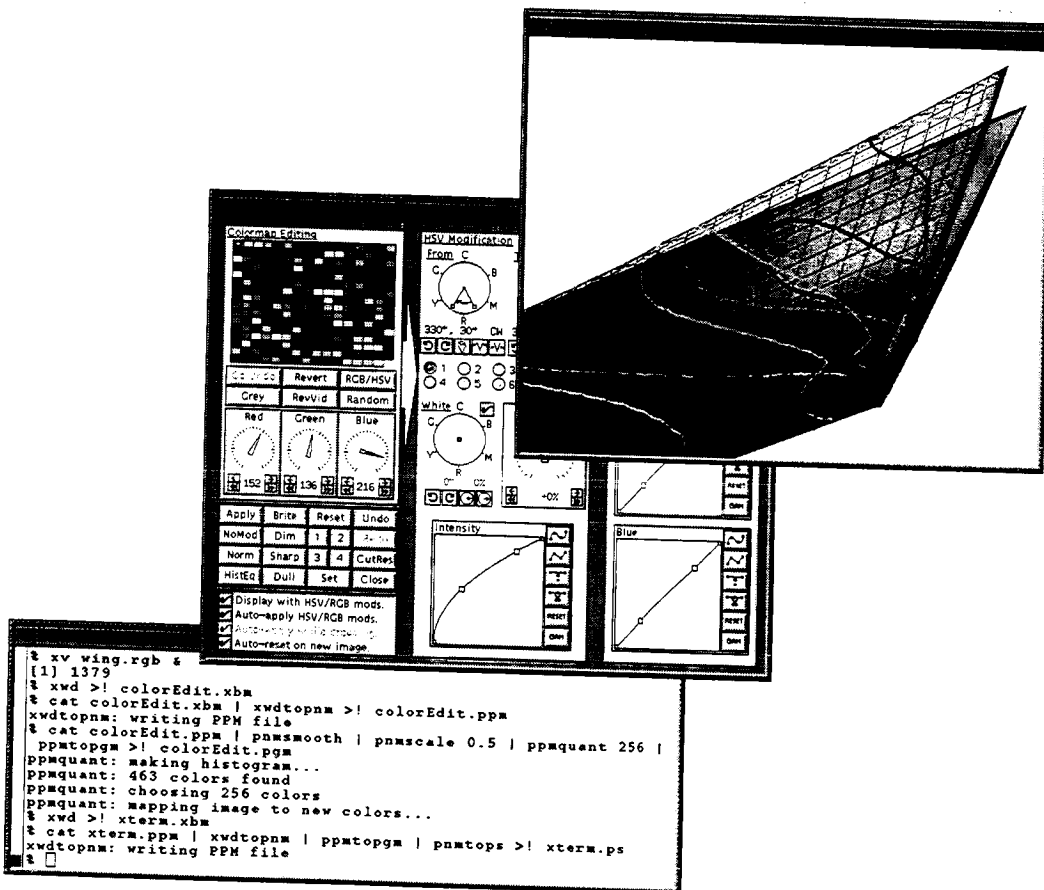
356027

P.

Compiled by
Carol D. Wieseman
Langley Research Center • Hampton, Virginia

N95-16453
--THRU--
N95-16480
Unclas

G3/62 0033238



(NASA-CP-10159) THE ROLE OF
COMPUTERS IN RESEARCH AND
DEVELOPMENT AT LANGLEY RESEARCH
CENTER (NASA, Langley Research
Center) 605 p

Proceedings of a workshop sponsored by the
National Aeronautics and Space Administration,
Washington, D.C., and held at
Langley Research Center, Hampton, Virginia
June 15-16, 1994

National Aeronautics and Space Administration
Langley Research Center • Hampton, Virginia 23681-0001

October 1994

INTRODUCTION

On June 15 - 16, 1994, the Computer Systems Technical Committee presented a workshop, "The Role of Computers in LARC R&D", at NASA Langley Research Center. The objectives of the 1994 Workshop were to inform the LARC community about the current software system and software practices being used at LARC. To meet these objectives, there were talks presented by members of the Langley community, Naval Surface Warfare Center, Old Dominion University and Hampton University.

The workshop was organized in 10 sessions as follows:

- Software Engineering
- Software Engineering Standards, Methods, and CASE Tools
- Solutions of Equations
- Automatic Differentiation
- Mosaic and the World Wide Web
- Graphics & Image Processing
- System Design and Integration
- CAE Tools
- Languages
- Advanced Topics

This document is a compilation of the presentations given at the workshop. The Conference was also videotaped and the videotapes are archived at the NASA Learning Resource Center (804-864-2325).

Appreciation is expressed to the individuals that participated by presenting and attending the workshop.

Norma Campbell, CSTC co-chair

PROCEEDINGS OF THE WORKSHOP ON ROLE OF COMPUTERS IN
LaRC RESEARCH AND DEVELOPMENT

INTRODUCTION	i
PARTICIPANTS	v
LaRC COMPUTER SYSTEMS TECHNICAL COMMITTEE	viii
<u>SESSION 1 Opening Session</u>	1
- <u>Chaired by Jerry H. Tucker</u>	
1.1 RTG Perspectives on Computing at LaRC	2
- Doug Dwoyer	
1.2 IOG Perspectives on Computing at Langley	10
- Frank Allario	
<u>SESSION 2 Software Engineering</u>	20
- <u>Chaired by Susan J. Voigt</u>	
2.1 Software Engineering from a Langley Perspective.....	21
- Susan Voigt	
2.2 Panel on Perspectives on Software Development	43
- Chuck Niles, Pam Rinsland, Pat Schuler, Peg Snyder, Tom Zang, Brenda Zettervall	
<u>SESSION 3 Software Engineering Standards, Methods, and CASE Tools</u>	67
- <u>Chaired by Susan Voigt</u>	
3.1 Model-based Software Process Improvement	68
- Brenda Zettervall	
3.2 A Study of Software Standards Used in the Avionics Industry	85
- Kelly Hayhurst	
3.3 A Software Tool for Dataflow Graph Scheduling	106
- Robert Jones	
3.4 Use of Software Through Pictures on CERES.....	114
- Troy Anselmo	
<u>SESSION 4 Solutions of Equations</u>	129
- <u>Chaired by Olaf Storaasli</u>	
4.1 Rapid Solution of Large-scale Systems Of Equations	130
- Olaf Storaasli	
4.2 Solution of Matrix Equations Using Sparse Techniques	147
- Majdi Baddourah	
4.3 Equation Solvers for Distributed Memory Computers	156
- Olaf Storaasli	
<u>SESSION 5 Automatic Differentiation</u>	167
- <u>Chaired by Olaf Storaasli</u>	

5.1	Applications of Automatic Differentiation in Computational Fluid Dynamics	168
	- Larry Green	
5.2	Automatic Differentiation for Design Sensitivity Analysis of Structural Systems Using Multiple Processors	181
	- Duc Nguyen, Olaf Storaasli, Jiangning Qin and Ramzi Qamar	
<u>SESSION 6 Mosaic and the World Wide Web.....</u>		212
<u>- Chaired by Clyde R. Gumbert and John W. McManus</u>		
6.1	Introduction to the World Wide Web and Mosaic	213
	-Jim Youngblood	
6.2	Use of World Wide Web and NCSA Mosaic at Langley	224
	-Michael Nelson	
6.3	How To Use the WWW To Distribute Scientific & Technical Information (STI)	237
	-Donna Roper	
<u>SESSION 7 Graphics and Image Processing</u>		246
<u>- Chaired by David C. Banks</u>		
7.1	Image Tools for UNIX	247
	- David Banks	
7.2	From Computer Images To Video Presentation: Enhancing Technology Transfer	266
	- Sheri Beam	
7.3	Data Visualization and Animation Lab (DVAL) Overview	272
	- Bill Von Ofenheim , Kathy Stacy	
7.4	Data Visualization and Animation Lab: Applications	292
	- Kurt Severance and Mike Weisenborn	
<u>SESSION 8 System Design and Integration</u>		316
<u>- Chaired by Jerry H. Tucker</u>		
8.1	The Design Manager's Aid for Intelligent Decomposition DeMAID	317
	- Jim Rogers	
8.2	RDD-100 and the Systems Engineering Process	351
	- Robert Averill	
8.3	Computer Tools for Systems Engineering at LaRC.....	377
	- J. Milam Walters	
8.4	A Distributed Computing Environment for Multidisciplinary Design FIDO	385
	- Robert Weston	
8.5	An Overview of the Computer Aided Engineering and Design for Electronics Laboratory CAEDE.....	412
	- Shelley Stover	
8.6	The Software Engineering and/or Ada Lab (SEAL).....	429
	- Robert Kudlinski	
<u>SESSION 9 CAE Tools</u>		434
<u>- Chaired by Carol D. Wieseman</u>		
9.1	Digital Control of Wind-Tunnel Models Using LabVIEW	435
	- Sherwood T. Hoadley	

9.2 Electronic Engineering Notebook -A Software Environment for Research Execution, Documentation, and Dissemination	443
- Dan Moerder	
9.3 IDEAS ² Computer Aided Engineering Software.....	470
- Pat Troutman	
9.4 Matlab as a Robust Control Design Tool	485
- Irene Gregory	
9.5 Simulation of the Coupled Multi- Spacecraft Control Testbed at The Marshall Space Flight Center.....	497
- Dave Ghosh, and Raymond C. Montgomery	
<u>SESSION 10 Languages</u>	
- <u>Chaired by Robert F. Estes</u>	518
10.1 Object Oriented Numerical Computing in C++	519
- John Van Rosendale	
10.2 Hardware Description Languages	536
- Jerry H. Tucker	
10.3 High Performance FORTRAN.....	546
- Piyush Mehrotra	
<u>SESSION 11 Advanced Topics</u>	562
- <u>Chaired by Susan Voigt</u>	
11.1 Current research activities at the NASA-sponsored Illinois Computing Laboratory of Aerospace Systems and Software	563
- Kathryn Smith	
11.2 Epistemology, SoftwareEngineering, and Formal Methods	570
- C. Michael Holloway	

PARTICIPANTS

Frank Allario
Mail Stop 157
NASA Langley Research Center
Hampton, VA 23681-0001

Troy Anselmo
Science Applications Int., Corp.
Mail Stop 927
NASA Langley Research Center
Hampton, VA 23681-0001

Robert Averill
Mail Stop 430
NASA Langley Research Center
Hampton, VA 23681-0001

Majdi Baddourah
Lockheed Engineering & Sciences Co.
Mail Stop 240
NASA Langley Research Center
Hampton, VA 23681-0001

David C. Banks
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001

Sherilee F. Beam
Computer Sciences Corporation
Mail Stop 157B
NASA Langley Research Center
Hampton, VA 23681-0001

Douglas L. Dwoyer
Mail Stop 105
NASA Langley Research Center
Hampton, VA 23681-0001

Robert F. Estes
Mail Stop 288
NASA Langley Research Center
Hampton, VA 23681-0001

Dave Ghosh
c/o Ray Montgomery
Mail Stop 161
NASA Langley Research Center
Hampton, VA 23681-0001

Larry Green
Mail Stop 159
NASA Langley Research Center
Hampton, VA 23681-0001

Irene Gregory
Mail Stop 489
NASA Langley Research Center
Hampton, VA 23681-0001

Clyde R. Gumbert
Mail Stop 159
NASA Langley Research Center
Hampton, VA 23681-0001

Kelly Hayhurst
Mail Stop 130
NASA Langley Research Center
Hampton, VA 23681-0001

Sherwood T. Hoadley
Mail Stop 340
NASA Langley Research Center
Hampton, VA 23681-0001

C. Michael Holloway
Mail Stop 130
NASA Langley Research Center
Hampton, VA 23681-0001

Robert Jones
Mail Stop 473
NASA Langley Research Center
Hampton, VA 23681-0001

Robert Kudlinski
Mail Stop 157
NASA Langley Research Center
Hampton, VA 23681-0001

John W. McManus
Mail Stop 125B
NASA Langley Research Center
Hampton, VA 23681-0001

Piyush Mehrotra
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001

Dan Moerder
Mail Stop 161
NASA Langley Research Center
Hampton, VA 23681-0001

Raymond C. Montgomery
Mail Stop 161
NASA Langley Research Center
Hampton, VA 23681-0001

Michael Nelson
Mail Stop 157A
NASA Langley Research Center
Hampton, VA 23681-0001

Duc Nguyen
Old Dominion University
c/o Olaf Storaasli

Chuck Niles
Mail Stop 442
NASA Langley Research Center
Hampton, VA 23681-0001

Pam Rinsland
Mail Stop 472
NASA Langley Research Center
Hampton, VA 23681-0001

Jim Rogers
Mail Stop 246
NASA Langley Research Center
Hampton, VA 23681-0001

Donna Roper
Mail Stop 444
NASA Langley Research Center
Hampton, VA 23681-0001

Pat Schuler
Mail Stop 125A
NASA Langley Research Center
Hampton, VA 23681-0001

Kurt Severance
Mail Stop 125A
NASA Langley Research Center
Hampton, VA 23681-0001

Kathryn Smith
Mail Stop 478
NASA Langley Research Center
Hampton, VA 23681-0001

Peg Snyder , retired
Mail Stop 111
NASA Langley Research Center
Hampton, VA 23681-0001

Kathy Stacy
Mail Stop 125A
NASA Langley Research Center
Hampton, VA 23681-0001

Olaf Storaasli
Mail Stop 240
NASA Langley Research Center
Hampton, VA 23681-0001

Shelley Stover
Mail Stop 488
NASA Langley Research Center
Hampton, VA 23681-0001

Pat Troutman
Mail Stop 288
NASA Langley Research Center
Hampton, VA 23681-0001

Jerry H. Tucker
Mail Stop 488
NASA Langley Research Center
Hampton, VA 23681-0001

Susan J. Voigt
Mail Stop 288
NASA Langley Research Center
Hampton, VA 23681-0001

John Van Rosendale
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001

Bill Von Ofenheim
MS 125A
NASA Langley Research Center
Hampton, VA 23681-0001

J. Milam Walters
Mail Stop 430
NASA Langley Research Center
Hampton, VA 23681-0001

Mike Weisenborn
Mail Stop 125A
NASA Langley Research Center
Hampton, VA 23681-0001

Robert Weston
Mail Stop 159
NASA Langley Research Center
Hampton, VA 23681-0001

Carol D. Wieseman
Mail Stop 340
NASA Langley Research Center
Hampton, VA 23681

Jim Youngblood
Lockheed Engineering & Sciences Co.
Mail Stop 904
NASA Langley Research Center
Hampton, VA 23681

Tom Zang
Mail Stop 159
NASA Langley Research Center
Hampton, VA 23681

Brenda Zettervall
Code 6000A1
Port Hueneme Division
Naval Surface Warfare Center
1920 Regulus Ave.
Virginia Beach, VA 23461-2097

LaRC COMPUTER SYSTEMS TECHNICAL COMMITTEE

The LaRC Computer Systems Technical committee was established in 1991 by the Chief Scientist of Langley Research Center, Michael F. Card. The goal of this Committee is to foster the exchange of technical information between the various groups who are involved in R&D on computer systems. The technical committee endeavors to provide workshops, luncheon speakers (brown bag seminars), and outside experts. The technical committee strives to provide technical interchange on computer systems.

The current membership of the CSTC consists of the following personnel:

TUCKER, JERRY H, chairman

Mail Stop 488
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-7342
email: J.H.Tucker@LaRC.nasa.gov

GUMBERT, CLYDE R

Mail Stop 159
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-2221
email: C.R.Gumbert@LaRC.nasa.gov

CAMPBELL, NORMA K, cochairman

Mail Stop 355
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-1131
email: N.K.Campbell@LaRC.nasa.gov

MCINTOSH, LEE T (TOM)

Mail Stop 236
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-4676

WIESEMAN, CAROL D, secretary

Mail Stop 340
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-2824
email: C.D.Wieseman@LaRC.nasa.gov

MCMANUS, JOHN W

Mail Stop 125B
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-4037
email: J.W.Mcmanus@LaRC.nasa.gov

CARPENTER, CHUCK L

Mail Stop 125A
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-8046
email: C.L.Carpenter@LaRC.nasa.gov

RUGGLES, STEPHEN L

Mail Stop 473
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-1515

FOX, CHARLES H, JR

Mail Stop 361
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-4906
email: C.H.Fox@LaRC.nasa.gov

TRUSSELL, PHILLIP T

Mail Stop 442
NASA Langley Research Center
Hampton, VA 23681
Phone: 804 864-6961
email: P.T.Trussell@LaRC.nasa.gov

SESSION 1 Opening Session

Chaired by

Jerry H. Tucker

- 1.1 RTG Perspectives on Computing at LaRC - Doug Dwoyer**
- 1.2 IOG Perspectives on Computing at Langley - Frank Allario**



Langley Research Center
Research & Technology Group

RTG Perspectives on Computing at LaRC

June 15, 1994

Doug Dwoyer

**“Three technologies are revolutionizing
our world:
silicon chips,
light fibers, and
software”**

**Dr. John Mayo
President, Bell Labs**



Langley Research Center
Research & Technology Group



Two RTG Perspectives on Computing at LaRC

- Impact on how we do our research
- Impact on what research we do



Langley Research Center
Research & Technology Group



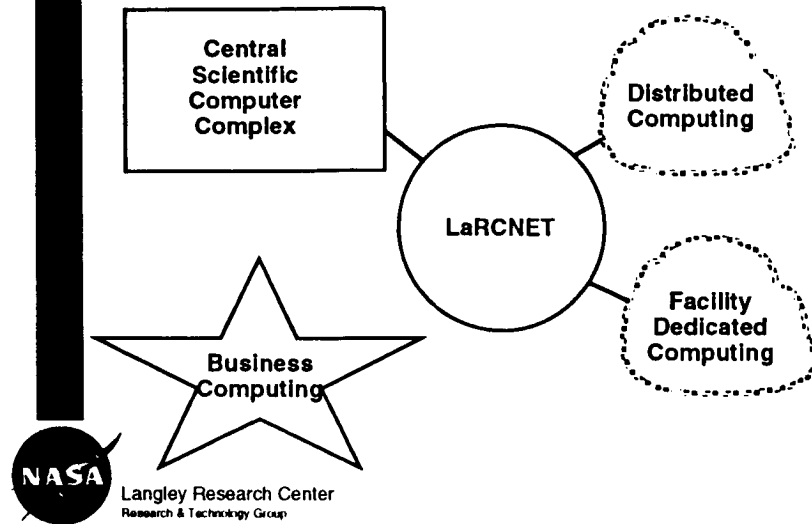
Two RTG Perspectives on Computing at LaRC

- 
- Impact on what research we do



Langley Research Center
Research & Technology Group

The Computing Universes at Langley



Future Computing Universe of Langley



RTG ADP n-Team

- **Membership:** RTG division ADP managers & at-large ADP personnel
- **Role:**
 - Provide advice and consultation to RTG line management on internal and external ADP issues and resource requirements
 - Represent the RTG's position as "ADP customer" to IOG for long-range planning of ADP services
 - Provide effective and cost-efficient planning, implementation, operation, and upgrade of distributed computing services within the RTG
- **Audience:** RTG line management & computer users; ADP providers (ISD, ETTD)



Langley Research Center
Research & Technology Group

Future Computing Universe of Langley--Expected Outcomes

- Enable exploration of unknown
- Universal communication
- Research productivity
- Removal of distance
- Support conversion of information into knowledge
- Fundamentally change our research products?



Langley Research Center
Research & Technology Group



Future Computing Universe of Langley--Expected Properties

- **Computational environment that serves individual user needs**
- **Integrated information management capability**
- **Harmonious relationship of mini-environments vs. unified environment**
- **Reuse of previously developed assets**
- **Software engineering standards and techniques widely applied**



Langley Research Center
Research & Technology Group



Two RTG Perspectives on Computing at LaRC

- **Impact on how we do our research**
- 



Langley Research Center
Research & Technology Group

Impact on what research we do-- Transportation Research

- **Requirement for transportation**
 - Impact of the revolutionizing technologies?



Langley Research Center
Research & Technology Group

Impact on what research we do-- Transportation Research

- **Impact of three revolutionizing technologies on aero and space transportation**
 - National Airspace System
 - on-board computing
 - cockpit automation
 - design/manufacture/field support
- **NASA impact not clear**
 - develop the technology and they will come doesn't work
 - 3rd generation research thinking required



Langley Research Center
Research & Technology Group




Impact on what research we do-- Atmospheric Science

- **Creation and storage of more and more information**
- **Conversion of vast storehouses of information into knowledge**
- **Creative use of information technologies is critical**



Langley Research Center
Research & Technology Group



Impact on what research we do-- Systems Analysis

- **Must utilize level of technology industry uses to remain credible**



Langley Research Center
Research & Technology Group

Concluding Remarks

- Future information technologies will have profound effects on Langley
- We must learn how to positively take advantage of them



Langley Research Center
Research & Technology Group

INFORMATION SYSTEMS DIVISION

**ISD PERSPECTIVE ON ...
“ COMPUTING at Langley ”**

Dr. Frank Allario

June 15, 1994

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

• **OUTLINE ...**

- GENERAL OBSERVATIONS ON THE FOLLOWING >>>
 - » SUPERCOMPUTING WITHIN OA ...
 - » INFORMATION SERVICES ... “ What are they? ”
 - » RAPID PROTOTYPING ... “ The wave of the future! ”
- ISD STRATEGIC THRUSTS as presented to the ...
 - » LCUC ...
 - » RTG / ADP “n” Team ...
- SOME KEY ISD ACCOMPLISHMENTS as they relate to NASA Langley and National priorities ...
- SUMMARY COMMENTS on the future of Computing ...

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

- **GENERAL OBSERVATIONS ON THE FOLLOWING >>>**
 - » **SUPERCOMPUTING WITHIN OA ...**
 - Langley and the other OA Centers will depend more upon a **Centralized site** for providing high capacity, mass storage capability ...
 - Langley will depend **exclusively** upon its high bandwidth, **networking capability** to communicate with the outside world, and provide supercomputing capability for computationally intensive calculations ...
 - A "**clustered architecture**" will be developed at Langley & other OA Centers to handle **mid-range computational requirements** ...
 - This "clustered architecture" should become a **joint venture** between IOG / ISD, RTG / ADP and RTG / HPCCP ...
 - Langley will eventually **outsource all large scale computing** ...
 - ISD will focus its Human Resources upon **other Information Services** ...

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

- **SUPERCOMPUTING WITHIN OA & Langley ...**
 - **LANGLEY / ISD STRENGTHS FOR OUR CUSTOMERS INCLUDE THE FOLLOWING ...**
 - » **A GOOD GRASP ON MASS STORAGE TECHNOLOGY ...**
 - » **A LEAD IN HIGH BANDWIDTH NETWORKING TECHNOLOGY ...**
 - » **A GOOD HANDLE ON NATIONAL & AGENCY GOALS IN THE MASSIVELY PARALLEL PROCESSING [MPP] TECHNOLOGY ...**
 - **LANGLEY WILL RECEIVE AN IBM, SP-2 IN AUGUST 1994 FOR INTERNAL EVALUATION & CONSULTATION TO INDUSTRY ...**

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

• SUPERCOMPUTING WITHIN OA & LANGLEY ...

- While you and I sleep silently at night, we have a corps of system engineers and technicians who keep our facilities "purring," in our Central Scientific site, our Communications Systems, our Specialized Laboratories, and our Flight Simulation Facilities ...
- During severe storms, power outages, and hurricanes our corps of contractors and civil servants keep the vitality of the Center's research mission alive, through dedicated engineering support ...
- As a new boy on the block in computing, I am taking the time to say ... " Thanks from all of us !"

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

• INFORMATION SERVICES ... " What are they? "

- SCIENTIFIC SUPPORT SERVICES ...
 - » **SPECIALIZED LABORATORIES ...**
 - EOS / DAAC ...
 - FLIGHT SIMULATION ...
 - GEOLAB ...
 - DVAL ...
 - SEAL ...
 - » **SPECIALIZED INFORMATION SERVICES ...**
 - DATA MANAGEMENT ARCHITECTURES ...
 - SOFTWARE ENGINEERING ...
 - HIGH BANDWIDTH, NETWORKING SYSTEMS ...
 - TRAINING IN INFORMATION SYSTEMS & TECHNOLOGY...

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

- **INFORMATION SERVICES ... " What are they? "**
 - TECHNICAL SUPPORT SERVICES ...
 - » MANAGEMENT INFORMATION SYSTEMS ...
 - » ELECTRONIC COMMUNICATION SYSTEMS ...
 - » BUSINESS ACCOUNTING SYSTEMS ...
 - » COMPUTER SECURITY SYSTEMS ...
 - » OFFICE MANAGEMENT TRAINING ...
 - » SECURE NETWORKING SYSTEMS ...
 - » HELP DESKS & TRAINING SERVICES ...
 - » VOICE / VIDEO / VISUAL TECHNICAL TOOLS ...

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

- **RAPID PROTOTYPING ...**
 - " The wave of the future! "**
 - I AM ENCOURAGING ALL PERSONNEL WITHIN ISD, where it is reasonable, TO CONDUCT RAPID PROTOTYPING TO INSURE WE PROVIDE OUR CUSTOMERS FULL UNDERSTANDING OF THE ...
 - COSTS OF DOING BUSINESS...**
 - "What you really need, versus what we as a National Laboratory, really can afford!"
 - I WOULD APPRECIATE YOUR THOUGHTS ON E-Mail, WHICH COULD HELP ISD UNDERSTAND THIS CONCEPT ...

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

**–SUMMARY COMMENTS, on the future of
Computing @ Langley ...**

- » **THE RESEARCH PROCESS WILL CHANGE AND WITH THAT CHANGE THE ROLE OF DVAL WILL BECOME INDISPENSIBLE TO THE RESEARCH PROCESS ... { ISD + STID + ETTD = ONE }!**
- » **Langley / Ames / Lewis will consolidate NASA resources in specialized, scientific services for our researchers + industry partners and make them available nationally to the Aeronautics community ...**
- » **Langley will lead the software engineering process for applications to Aeronautics ...**

**ISD's PERSPECTIVE ON COMPUTING
@ Langley ...**

**–SUMMARY COMMENTS on the future of
Computing @ Langley ...**

- » **THE CENTRAL SCIENTIFIC COMPUTER COMPLEXES, WILL BE CONSOLIDATED INTO ...
“ META-CENTER CONCEPTS... ”**
- » **LANGLEY WILL LEAD OA CENTERS IN THIS CONCEPT ...**
- » **OUR FUTURE IS BRIGHT, BUT WE MUST ESTABLISH SCIENTIFIC PRIORITIES WITHIN OUR SCIENTIFIC CORPS, IN AERONAUTICS + SPACE SCIENCES COMMUNITIES ...**
- » **OR ELSE ...**

Information Systems Division

Overview of ISD

- ISD was formed through the merger of ACD, BDSO, and IRMO to provide a focal point for information services at LaRC
- Our mission is to lead the application of advanced information systems technologies that will improve the productivity and quality of the LaRC's processes and products
- The role of ISD is changing:
 - » Decreasing emphasis on providing central computing resources
 - » Increasing services and technologies to make researchers, managers, and distributed computing users more productive and effective
- Major "business areas" include communications, advanced technology computing, integrated computing environment, information resources management, management information systems, simulation systems, data management, visualization & analysis, and software engineering

Information Systems Division

ISD Planning

- Langley requirements (known and anticipated) and perceived trends were used to identify the major "business areas"
- These introductory presentations begin a continuous process of customer interactions to help determine ISD priorities, expectations, and future directions
- ISD is committed to customer satisfaction and we must work together to develop realistic expectations that provide exceptional service in a continually changing environment
 - » Growing demand for information systems and services
 - » Decreasing budgets and manpower
 - » Rapidly evolving technologies

We can do anything, but we can't do everything

Information Systems Division

Communications

Provide Center-wide coordinated, reliable, state-of-the-art, and cost effective voice, data, and video communication networks and services

- Integrate evolving advanced network technology, such as FDDI and ATM, to accommodate high performance distributed computing at LaRC
- Provide enhanced user access to the National Information Superhighway and world-wide information resources
- Lead deployment of portable and desktop video conferencing
- Lead the Center-wide E-mail integration initiative

Information Systems Division

Advanced Technology Computing

Provide effective scientific computing resources through implementation and support of evolving hardware and system software technology

- Utilize centralized, homogeneous workstation clusters to meet medium-scale computing requirements
- Employ local and remote vector supercomputers to meet large-scale computing requirements
- Evaluate high performance scaleable processing technology
- Implement Distributed Mass Storage System (DMSS) to meet rapidly increasing on-line and archival requirements; evaluate emerging mass storage technologies
- Apply tools, performance analyses, and code enhancements to improve the utilization of computing resources

Information Systems Division

Integrated Computing Environment

Lead the integration of central and distributed computing resources into a unified, cost-effective computing and communications environment

- Develop a uniform user environment and global file system
- Improve coordination and cost-effectiveness of system administration
- Lead Center-wide implementation of "Electronic Office" hardware, software, communications, and services
- Implement networked distribution and installation of common software packages
- Lead architectural design efforts to implement integrated Code R computing environment

Information Systems Division

Information Resources Management

Lead in the planning, acquisition, implementation and efficient management of Langley's information processing resources throughout their life-cycle

- Develop policies that satisfy ADP legislation and provide essential oversight with a minimum of "control"
- Implement the LaRC Computer Security Program
- Contain costs through strategic planning and consolidation of distributed ADP requirements
 - » Resource reutilization and sharing
 - » Hardware and software mass-buys, site licenses, maintenance, and services
 - » Centerwide electronic bulletin board "NewsNET" and user services

Information Systems Division

Management Information Systems

Develop and maintain efficient and effective information systems for business and administrative functions

- Perform software development, maintenance, operations, and data management for business applications, such as payroll, T&A, and property management
- Perform business process analysis, personal computing technology assessment, and training
- Develop Center and Headquarters management information systems
- Implement Agency standard systems and integrate with LaRC unique systems

Information Systems Division

Simulation Systems

Provide state-of-art, cost-effective, simulation capabilities to support LaRC's research and focused technology programs

- Development and operation of simulation systems to accomplish research program goals
 - » Real-Time Computer Systems (FSCS/ARTSS)
 - » Advanced Visual Systems (ACGI/WIDE, SGI/ONYX)
 - » Flight Decks (HSR, B737/757) and Motion Platforms (CMF)
- Rapid and cost-effective development of real-time models and software applications
 - » Vehicles and Vehicle Control Systems (HSR, HARV, B737/757)
 - » Flight Management Systems (FMC, Nav)
 - » Atmospheric models (Windshears, Wake Vortex)
- Development and implementation of new technology
 - » Improve efficiency of software and hardware development processes
 - » Provide new capabilities necessary to conduct research programs

Information Systems Division

Scientific Data Management, Analysis & Visualization

Apply advanced methods and tools to effectively manage and utilize computational and experimental research data

- Develop effective data management techniques to support the rapidly changing research environment
 - » Greatly Increasing data volume and multi-disciplinary communities
 - » On-line multimedia access and technology transfer
- Data Visualization and Animation Lab (DVAL) develops and applies advanced techniques to visualize, analyze, and present scientific data
- Geometry Modeling and Grid Generation (GEOLAB) provides a Center resource to effectively meet the large and varied grid geometry requirements in scientific computation
- High-end, production input and output devices, such as the LISAR Digitizing System and large format color plotters
- LaRC's EOSDIS Distributed Active Archive Center is a pathfinder for Worldwide information dissemination

Information Systems Division

Software Engineering

Improve the cost-effectiveness, quality, and performance of software developed at LaRC

- Implement a modern software development process that optimizes cost, schedule, system performance and reliability through the Software Engineering and(or) Ada Lab (SEAL)
- Develop mission critical software systems for LaRC space and avionics flight projects
- Meet increasing demand to apply software engineering to other LaRC software development efforts
 - » Applications include NTF DAS, HSR, and TAP
 - » Perform training, consultation, and information dissemination services
 - » Reverse engineer research software products (i.e., CFD) before transfer to customers

SESSION 2 Software Engineering

Chaired by

Susan J. Voigt

- 2.1 Software Engineering from a Langley Perspective - Susan Voigt
- 2.2 Panel on Perspectives on Software Development- Chuck Niles, Pam Rinsland, Pat Schuler, Peg Snyder, Tom Zang, Brenda Zettervall

SOFTWARE ENGINEERING FROM A LANGLEY PERSPECTIVE

by Susan Voigt

P. 47

This presentation is intended to provide a brief introduction to software engineering to set the stage for the panel discussion and some of the workshop presentations.

The talk is organized into four sections, beginning with the question "What is Software Engineering?" followed by a brief history of the progression of software engineering at LaRC in the context of an expanding computing environment. Several basic concepts and terms are introduced, including software development life cycles and maturity levels. Finally, some comments are offered on what software engineering means for LaRC and where to find more information.

In an article in the ACM Computing Surveys in 1978 (Vol. 10, No. 2, p. 197), Marvin Zelkowitz defined software engineering as the "process of creating software systems." (Note: ACM is the Association for Computing Machinery.) The IEEE Standard 610.12-1990 (Standard Glossary of Software Engineering Terminology) has a widely accepted definition that effectively is the application of an engineering approach to software.

The term "software engineering" was used at NATO conferences in 1968 and 1969, but became commonplace in 1975 when the first national conference (which became international at the second) was held in Washington, D.C. In that same year, the IEEE began publishing the journal: IEEE Transactions on Software Engineering. NASA started funding software engineering research as part of the Computer Science Research Program in the Office of Aeronautics and Space Technology in 1983. The Department of Defense was also concerned with "the software problem" in this time frame, and in 1984 the Software Engineering Institute was established at the Carnegie Mellon University. NASA's Office of Safety and Mission Assurance (Code Q) established the NASA Software Engineering Program in 1991, with funding for and active participation from LaRC.

Just as software engineering was developing, our computing environment was becoming more dispersed. In the 1960s, computing was done by computing professionals in a "closed shop" environment. However, by the 1970s, FORTRAN was used by researchers across the Center, and they had access to the centrally located computer facility by using the "green tub" service for pick up and delivery of punched cards and printed output (also called computer listings). In the mid-1970s, microprocessors and time sharing came to LaRC, providing remote computing capability. Computing expanded in the 1980s with distributed systems, personal computers, and data acquisition and/or control systems in many facilities. The 1990s has brought even more powerful workstations and networked systems. This changing environment has decentralized the computing and software development at the Center, so that software is now created in many organizations, with little coordination or collaboration.

One of the fundamental concepts in software engineering is that of life cycle. The life cycle is a way to capture the schedule and discipline of key activities, reviews (such as system design, requirements review and design review), and deliverable items at specific points in time. The Department of Defense has identified three "program strategies" in their recent standards, that illustrate classic software life cycles: waterfall, incremental and spiral.

The Grand Design strategy assumes a complete definition of the requirements prior to design. The waterfall life cycle includes the development phases: requirements analysis, design, coding, test and integration and finally operations and maintenance. As each phase is completed, products are delivered that support the next phase.

The Incremental strategy is also called "preplanned product improvement.". The user needs and system requirements are defined followed by a phased development with several releases or system builds. Each phase includes the typical steps in the waterfall process. Experience with early releases in the incremental approach can provide refinements for subsequent releases, along with the new capabilities planned.

The Evolutionary strategy is based upon Barry Boehm's spiral model (described in ACM Software Engineering Notes, Vol. 11, No. 4, Aug. 1986, pp. 14-24; and IEEE Computer, May 1988, pp. 61-72). This approach encourages consideration of risks, constraints and alternatives. The software development occurs in the third quadrant of the spiral, and is similar to the incremental development.

The Software Productivity Consortium (Lockheed, one of our support service contractors, is a member company) has extended the spiral model into the Evolutionary Spiral Process (ESP) Model with extensive training and guidebook materials available to SPC members (and to NASA, as a Lockheed customer).

The Software Engineering Institute (SEI) has defined the Capability Maturity Model (CMM) that can be used to identify how an organization can improve the maturity of its software process. The CMM has five levels, from initial to optimizing. Watts Humphrey, SEI fellow, is considered the author of the CMM. We do have copies of SEI provided documentation on the CMM in the Space Systems and Concepts Division. The lowest level (1) is when software development is informal and each job is only as good as the individual software developer. This is the stage when good software results from heroic effort. Level 2, called "repeatable," is more intuitive, where there are some common practices, but problems invariably arise when something new is introduced into the process. The focus at level 2 is on project management. The "defined level" (3) is qualitative and focused on the engineering process. The process has been written down, and the organization has accepted it as common practice. Training in the process is available, providing continuity with personnel turnover, and the staff meets regularly to discuss improvements. The quantitative or "managed level" (4) has measures in place to track productivity. The focus is on both product and process quality. The process is understood and managed so that bottlenecks can be identified and automated tools can implement parts of the process to reduce human error. When an organization has achieved the "optimizing level" (5), detailed metrics on the process are collected, problems can be anticipated, there is constant process improvement, and new technology can be infused. A level 5 organization is practicing TQM in software development to the full extent. At the present time, most organizations are at level 1 or 2.

The SPA and SCE are two assessment methods defined by the SEI. SPA, the Software Process Assessment, is used by an organization to assess their own actual process maturity and develop a software process improvement strategy. It is only for internal use. SCE, the Software Capability Evaluation, is more like an audit. It is used to gather information on the software process maturity of organizations that might be competing for a software task. Several government agencies are using SCE's in their source selection process. Our panelist from the Naval Surface Warfare Center has been trained in the SCE, and she will share some insights on this later. An analogy to compare the SPA and SCE: An assessment is like having a friend or relative help you prepare your income taxes (it's internal), whereas an evaluation is like having the IRS do an audit of your taxes.

Some other basic concepts of software engineering can be introduced by defining some jargon. CASE (computer aided software engineering) is a generic term to describe tools and environments that provide automated support for software development. The DOD has used CSCI (computer software configuration item) to describe major software modules

(that are kept under configuration control). Submodules are called Computer Software Components (CSC) and often compilation units are called Computer Software Units (CSU). CM stands for configuration management, a process for identifying and for controlling release and change of software items. Object Oriented Design (OOD) and object oriented programming are an alternative approach to procedural-oriented software architecture, treating programs and data as objects. IV&V is Independent Verification and Validation, the testing of software functionality and validation against requirements performed by a team separate from the developers. Software Quality Assurance (SQA) is an activity performed throughout the life cycle to assure that requirements analysis, design, code, and the resulting product satisfy the software requirements.

Additional jargon includes SMAP, which was the Software Management and Assurance Program led by the NASA Office of the Chief Engineer and later the Office of Safety, Reliability, Maintainability, and Quality Assurance (Code Q) in the 1980s. The SMAP team included representatives from all NASA Centers, and they helped define the NASA software documentation standards that have evolved to NASA STD-2100-91. The SMAP has been replaced with the Software Engineering Program in the current Code Q, Office of Safety and Mission Assurance. DID stands for Data Item Description, the Department of Defense (DOD) term used for software documentation format, instructions and outline. For example, the DOD-STD-2167A describing the current Defense System Software Development standard, contains at least 16 DIDs. The DOD program Software Technology for Adaptable, Reliable Systems, called STARS, has been active for over 10 years, and is the focus of considerable effort in areas including Software Engineering Environment (SEE) and Software Reuse. Research into reusing software assets (e.g., design and code segments) has included identification of domains or classes of application areas with common aspects where reuse makes sense.

Since the daily work at LaRC relies on software more and more, and as more emphasis is placed on the transfer of technology (which includes our software products), there is a need to pay more attention to the engineering of our software. There are several resources available to people at the Center, including the Software Engineering and/or Ada Laboratory (SEAL) in the Information Systems Division, an Inter-Group N-Team on Software Productivity, Quality, and Reliability led by Robert Estes, and Internet access to many information resources. The recently formed Hampton Roads Software Process Improvement Network (HRSPIN) offers additional opportunity for professional development and information exchange with individuals from government, industry and academia interested in software improvement. The Technical Library (as well as many individuals) have several of the software journals of particular value to the software engineering specialist.

There are several standards that also are applicable, and these can prove useful in guiding a software process. Experienced software engineers at NASA Langley are willing to share their knowledge, and the SPQR N-Team provides them an opportunity to network and work together to improve the quality of software at the Center.

Software engineering techniques can improve the software products developed for and by LaRC. The panel represents several perspectives on software development, and these experienced software developers and managers are willing share some of their views on where we are and where we should be going.

SOFTWARE ENGINEERING

by

**Susan J. Voigt
Space Systems and Concepts Division, SASPG**

**Presented to
The Role of Computers in LaRC R&D Workshop**

June 15, 1994

Outline

- **What is Software Engineering?**
- **A Brief History from LaRC Perspective**
- **Introduction to Some Basic Concepts**
- **What does this mean for LaRC?**

What Is Software Engineering?

- **"Process of creating software systems"**
(Zelkowitz, 1978 Computing Surveys)
- **"The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"**
(i.e., the application of engineering approach to software)

IEEE Std 610.12-1990

A Brief History - LaRC Perspective

- 1968/69 "Software Engineering" used at NATO Conferences
- 1973 Structured Programming in vogue
- 1975 First National (Int'l) Conference on Software Engineering
IEEE Transactions on Software Engineering journal started
- 1983 NASA Computer Science Research Program funded
(with Software Engineering component)
- 1984 CMU/Software Engineering Institute Established by DOD
- 1991 NASA Code Q started Software Engineering Program

Progress Of Computing At LaRC

- **In 1960s & early 70s, Programming done in ACD (Closed Shop)**
- **1970s - FORTRAN used by researchers Green Tub service (Open Shop)**
- **Mid-70s - Time Sharing in Central Facility Microprocessors in Labs**
- **1980s - Distributed systems, PCs, and automated facilities**
- **1990s - Networked workstations, virtual systems**

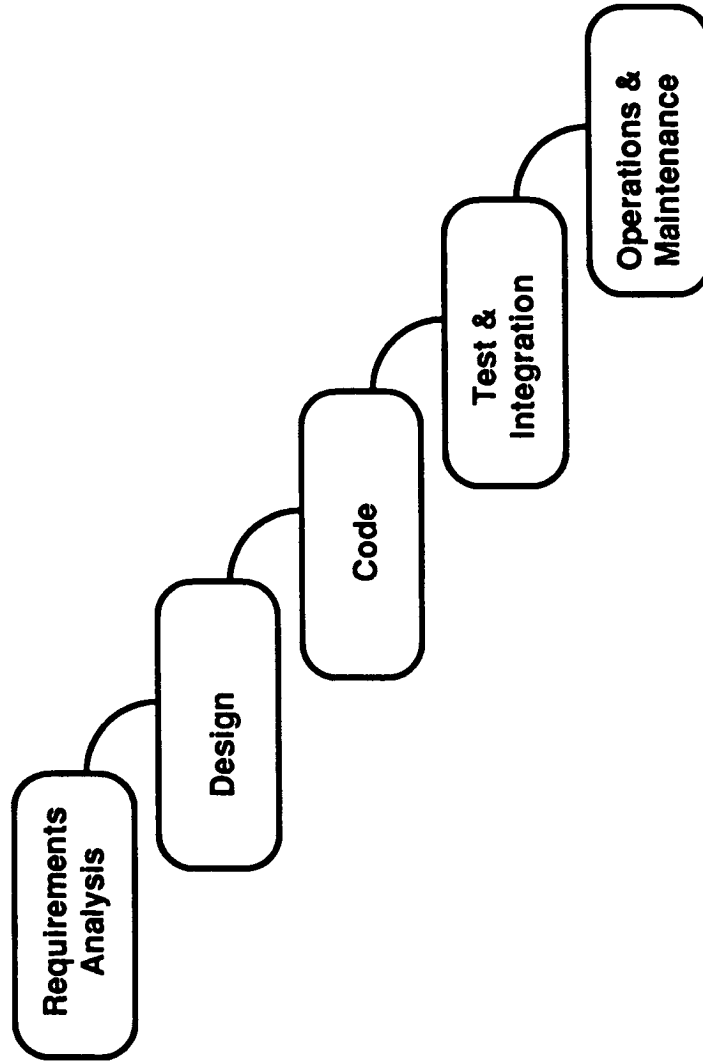
Software Life Cycles

Schedule of Activities, Key Reviews, and Deliverables

DoD Program Strategies

- **GRAND DESIGN** **Waterfall**
- **INCREMENTAL** **Preplanned Phased Development**
- **EVOLUTIONARY** **Spiral Process**

Waterfall



Incremental

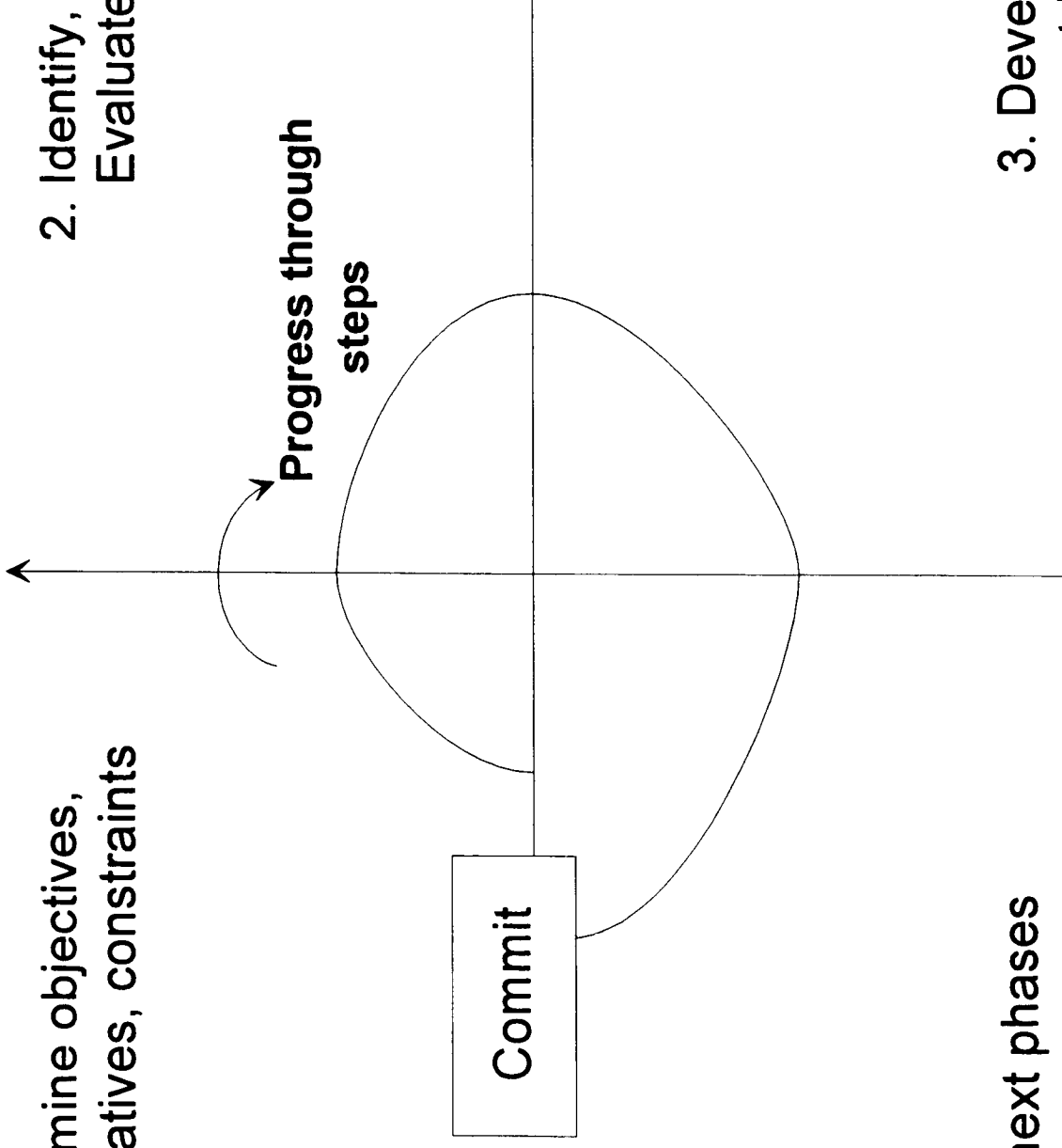


R - Requirements
D - Design
C - Code
T - Test & Integ.

Boehm's Spiral Lifecycle Model

1. Determine objectives, alternatives, constraints

2. Identify, resolve risks
Evaluate alternatives



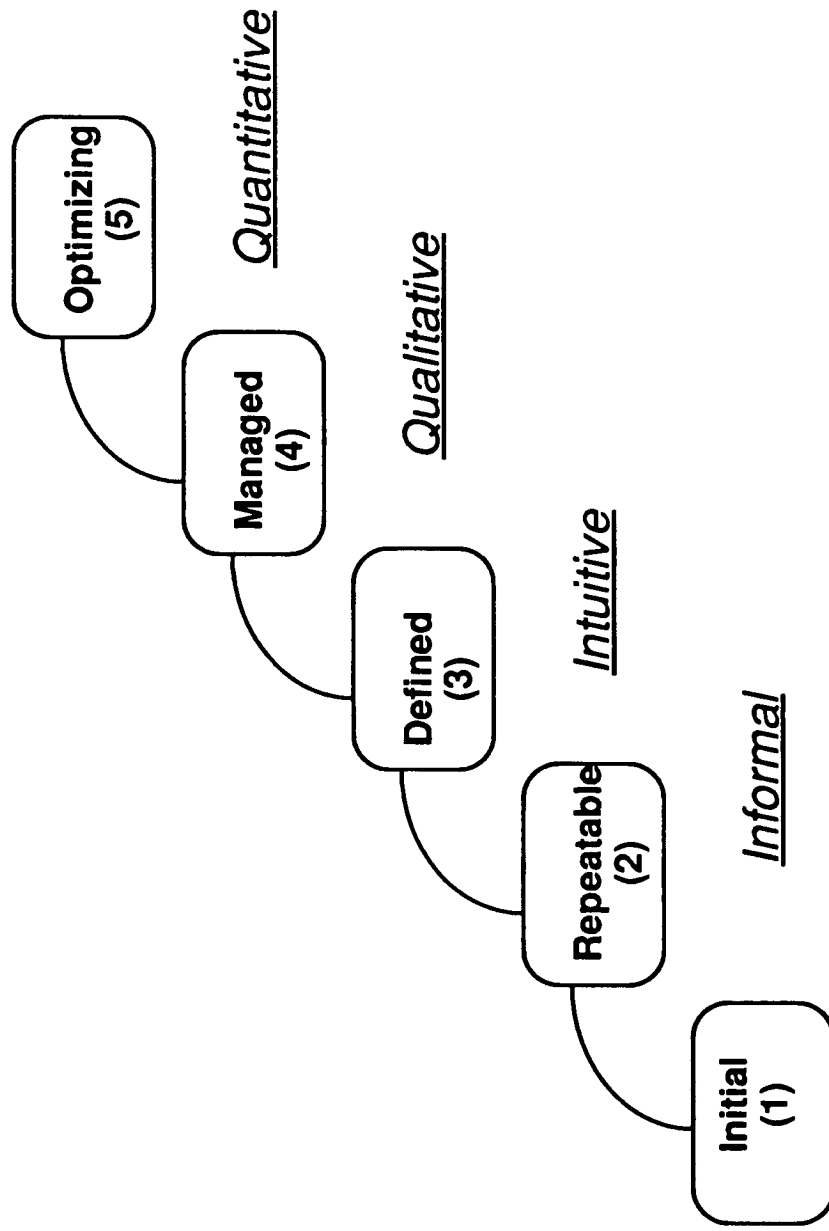
4. Plan next phases

3. Develop, verify
next-level product

Software Process Models

- **SPC** **Software Productivity Consortium**
(Lockheed is a member company)
- **ESP** **Evolutionary Spiral Process Model**
- **SEI** **Software Engineering Institute**
at Carnegie Mellon University
- **CMM** **Capability Maturity Model**
- **SPA** **Software Process Assessment**
- **SCE** **Software Capability Evaluations**

FIVE MATURITY LEVELS

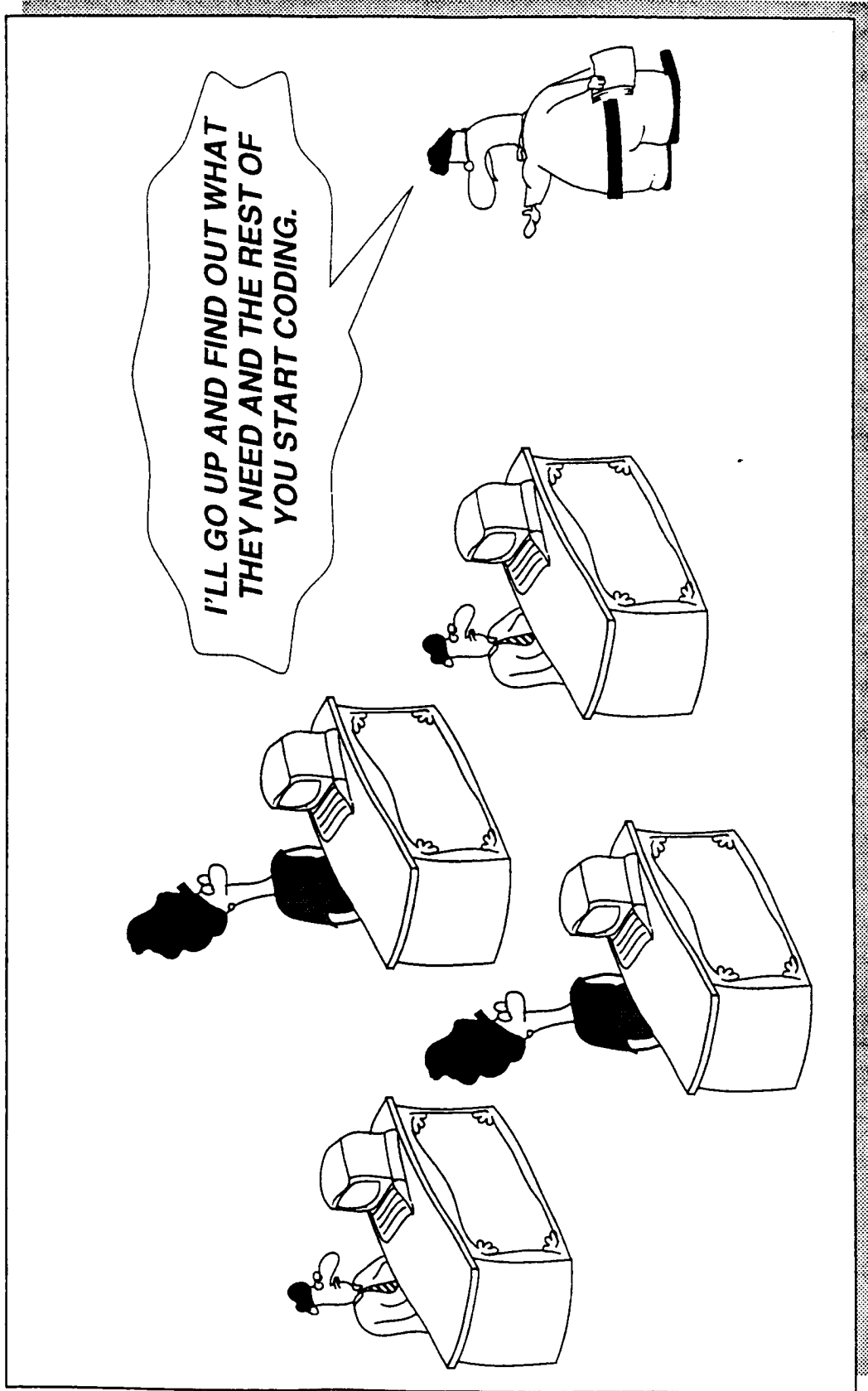


Capability Maturity Model

Level	Focus	Key Process Areas	Result
5 Optimizing	Continuous process improvement	Defect prevention Technology innovation Process change management	Productivity & Quality
	4 Managed	Product and process quality Process measurement and analysis Quality management	
3 Defined	Engineering process	Organization process focus Organization process defn. Peer reviews Training program Intergroup coordination Software product engineering Integrated software mgt.	
	2 Repeatable	Project management	
1 Initial	Heroes		



Characteristics of an Initial Level Organization



An Analogy

- **An *Assessment* is like asking your brother-in-law to help you prepare your income taxes**
- **An *Evaluation* is like having the IRS do an audit of your taxes**

Some Software Engineering Jargon

- **CASE** **Computer Aided Software Engineering**
- **CSCI/CSC/CSU** **Computer Software Configuration Item/Component/Unit**
- **CM** **Configuration Management**
- **OOD** **Object Oriented Design**
- **IV&V** **Independent Verification and Validation**
- **QA or SQA** **Software Quality Assurance**

More Software Engineering Jargon

- **SMAP** **NASA Software Management and Assurance Program (1980s)**
- **DID** **Data Item Description (Document Format)**
- **STARS** **DoD Software Technology for Adaptable, Reliable Systems Program (1984 - present)**
- **SEE** **Software Engineering Environment**
- **Software Reuse** **Incorporation of previously developed software products**
- **Domain** **Category or Application Area**

What Does This Mean For LaRC?

- **Increasing reliance on Software in Daily Work**
- **Technology Transfer Focus => Software Releases**
- **Several Activities and Resources are available**
 - ISD/SEAL (Software Engineering & Ada Lab)
 - SPQR N-Team
 - Mosaic/WWW Information
 - HRSPIN (Hampton Roads Software Process Improvement Network)
 - Journals (IEEE Software, IEEE Trans on SE, ACM SIGSOFT, ...)

Standards can be Resources

- **NASA-STD-2100** **Software Documentation Standard**
- **ISO 9001** **Quality System Standard**
A NASA Standard
- **ISO 9000-3** **Software Quality Guidelines**
- **DOD-STD-2167A** **Defense System Software Dev**
- **MIL-STD-498** **Software Development and Doc**
- **MIL-STD-499B** **Systems Engineering**
- **many IEEE Standards**

SUMMARY

- **Software Engineering is necessary for better software products**
- **Experienced Software Engineers do work at LaRC**
- **We need to improve our software quality in the spirit of "faster, better, cheaper"**
- **Join the LaRC Software Productivity, Quality, and Reliability N-Team !**

Summary of Panel on Perspectives on Software Development

The panel consisted of five NASA Langley employees representing different application domains and a representative from the Naval Surface Warfare Center in Virginia Beach, VA. Each panelist began with a short statement reflecting both experiences and perspectives on software development. The panelists, their application domain area, and organization were:

Chuck Niles	Facilities Software, IOG
Pat Schuler	Flight Software, IOG
Tom Zang	Researcher Software, RTG and LCUC Chair
Pam Rinsland	Embedded Systems Software, IOG
Peg Snyder	Science Software, SASPG (retired)
Brenda Zettervall	Software Quality Improvement, Naval Surface Warfare Center
Susan Voigt	Moderator, SASPG

Chuck Niles is in the Electrical and Electronic Systems Branch in the Facilities Systems Engineering Division of the Internal Operations Group. He has 15 years of experience in software development for wind tunnel control systems, process monitoring, and ground facilities communications on minicomputers and the whole family of Intel microprocessors. He is responsible for software configuration management for many wind tunnels at LaRC and has developed documentation for all phases of software development. His opening remarks, "Perspectives on Software Development," are included following this section.

Pat Schuler is in the Advanced Computer Systems Branch in the Information Systems Division of the Internal Operations Group. She began her Langley career providing support for scientific research computer applications. She was software manager for the first embedded systems (flight software) project in the Software Engineering and Ada Laboratory (SEAL). Since the SEAL was formed, she has been active in developing it as a center of excellence in software engineering at LaRC, with support from the NASA Office of Safety and Mission Assurance (Code Q). In this discussion, Pat represented the flight software for Langley scientific instruments.

Pat cited three characteristics of flight software development: embedded systems, distributed processing, and real-time. She went on to clarify these as follows:

Embedded systems - A specialized computer with custom-programmed software used to control functions within the device it's controlling.

Distributed processing - A system in which tasks to be performed by the available computing resources are executed by a number of processors, often in parallel.

Real-time - Results are calculated in sufficient time to guide the physical process under control.

She cited four typical examples of space flight projects at LaRC: CERES, JADE, LITE, and MIDAS with flight life-times ranging from 11 days to a few months to 5 years, and flight code size ranging from 2K to 18K (where K represents 1000 source lines of code). In addition to on-board flight software, ground support software, including simulators, test subsystems and mission operations subsystems must be developed, and these range from 2 to 10 times the size of the flight code. The SEAL has standardized on Microsoft Windows and other MS software, Ada, object-oriented design, formal inspections, and Novell as their local area network for internal mail and a shared group calendar. The SEAL tools, based on PC and Intel, are considered a Center resource. The SEAL is also trying to baseline their software development process and document it in guidebooks. They also are collecting metrics on how software is developed in the SEAL. SEAL personnel provide consultation to and arrange training for other groups at the Center in software engineering processes and tools, but they do have a limited staff. A list of the tools and software documents available from SEAL follows this section. Anyone interested in learning about the tools, their use, and related training should contact her.

Tom Zang is head of the Multidisciplinary Design Optimization Branch in the Fluid Mechanics and Acoustics Division of the Research and Technology Group. He also is the chair of the Langley Computer Users Committee (LCUC). He represented researcher software on the panel.

Tom said that the LCUC intends to reorganize itself in the fall to align with the new Center organization. The LCUC was set up about 20 years ago to provide a voice for user concerns and desires to the Analysis and Computation Division for short-term tactical and some long-term strategic planning.

The two products from research are reports and software. However, managers and researchers simply do not recognize the importance of their software as a technical product. He observed that NASA encourages the quality aspects of technical reports, but not of software. Four types of software products are produced by researchers at LaRC: concepts, portable modules, pilot codes, and production codes. The concepts may include new algorithms and these are published in technical reports. Modules are usually available as commented code. Pilots are prototype software for early release with caveats since it is not thoroughly tested, still may be in development, and has little documentation. Production codes are well written and well documented computer programs. A good example of multidisciplinary code at LaRC is FIDO (described by Bob Weston at a later session at the workshop). In closing Tom stated he would like to see management place greater value on good software, researchers write their

software for others as well as themselves, and software engineers act as a resource for others at LaRC. He did note that software engineering is included on the list of necessary skills in the Research and Technology Group (RTG).

In these proceedings, he has included a few charts from the LCUC files of a 1980 briefing by Jarek Sobieski which cite some of the same issues. A copy of Tom's transparencies "Perspectives on Software Development" are included following this section.

Pam Rinsland is the assistant head of the Electronics Systems Branch in the Aerospace Electronics Systems Division of the Internal Operations Group. In her 22 years at LaRC she experienced the transition from the batch-oriented central computing and plotting without preview to the "instant gratification" of time-sharing. She has developed software for a wide range of aerospace applications, including writing code to execute on computers ranging from Intel's first 4-bit processor to the first supercomputers delivered to LaRC. In her current position, she is in a hardware-oriented branch and promotes her firm beliefs in the absolute necessity of close ties between hardware and software specialists, and in maintaining discipline in the software development process.

Her opening remarks, Reflections from a "Jurassic Programmer" on Software Development at LaRC, follow this section.

Peg Snyder, prior to her retirement from NASA in May 1994, was in the Data Management Office in the Atmospheric Sciences Division of the Space and Atmospheric Sciences Program Group. She has 31 years of experience in software development at NASA, starting at Lewis Research Center with FORTRAN code on a mainframe with 30K of 36-bit words (memory) for basic research in nuclear physics scattering analysis and non-steady fluid flow. An early lesson she learned was to number your punched cards (artifacts now found in the museums in Washington, DC). She worked for several years in the Space Station Freedom Program Office prior to coming to LaRC 3 years ago. Her software experience ranges from office automation software and space applications to wind tunnel applications, data reduction, and CERES data processing.

Peg's most important message to the audience was that best results are obtained when an engineering approach is applied in the development of software. Specifically, her approach has six steps: 1) Define the problem (in the 1960s an engineer would bring a notebook to the programmer with "requirements" documented); 2) Figure out how to solve (reformulate the problem in terms of mathematics and select appropriate numerical analysis techniques); 3) Design the solution; 4) Implement the solution; 5) Test; 6) Use and maintain. We actually practiced more software engineering back in the batch days than we do

now. A second important message was that automated tools are only useful if they help you implement a process already in place.

Brenda Zettervall is Quality Improvement Administrator for East Coast Operations of the Port Hueneme Division of the Naval Surface Warfare Center located at Dam Neck in Virginia Beach, Virginia. She has 18 years of experience in software development including land-based integrated combat simulation programs and systems engineering necessary to translate operational requirements into simulation performance requirements. She is a member of the Software Engineering Institute (SEI) Capability Maturity Model (CMM) Advisory Board and the CMM Based Appraisal Review Group. She also is qualified to perform Software Capability Evaluations. She is the first chair of the recently formed Hampton Roads Software Process Improvement Network (HRSPIN).

Three years ago, Brenda became involved in quality improvement as part of a competition between Naval support centers and between the Navy and AF for software post-deployment support. Since the Navy is down-sizing and decommissioning many ships, software process improvement was necessary for survival since most of the systems supported at Dam Neck were on the "hit list." Being able to maintain cost and schedule is highly dependent on the maturity of the process in place. Hence they have embarked on establishing a management discipline for software development and maintenance. This means their process is documented, trained and enforced. The Navy is challenged to survive and to improve their software engineering process, since the Air Force has a vision to do all software engineering for the Department of Defense.

Questions from the Audience and Panelist Answers:

Q) Other engineering disciplines are based on mathematics. What is the basic science on which software engineering is based?

A) Mathematics is the basis for formal methods and algorithms such as rate monotonic scheduling.

Q) Suppose your organization were in charge of developing software for the next generation aircraft. Would you fly on it?

A) Four panelists said "Yes" and two said that flight critical software was outside their domain, and their organizations did not have the appropriate expertise and training.

Q) How will the software development process have changed 10 years from now?

A1) We will be doing it at home.

A2) Researchers will write code from day one using good practices - even if it is just "for themselves".

A3) We hope to raise our organization to higher maturity levels, hopefully close to a CMM level 5 and the Center to level 3 or 4.

A4) Necessity is the mother of invention. In the 60's there were incentives to make programs work smarter (e.g., you could be called in the middle of the night about your wind tunnel software if it didn't work properly). Things are changing, so we will be forced to be more rigorous.

A5) There will be a trend toward graphical programming models, and off-the-shelf packages available for control systems. There will be "6th generation programming languages".

A6) We will be rewarding people for good software engineering practices (activity will not be confused with productivity).

A7) Rapid prototyping and workstation platforms will be common.

Q) Where is a good place for Software Engineering at LaRC? In an N-team, a Branch or a Group?

A1) Software people are throughout the Center and there is no central focus or mechanism for software developers to exchange ideas and information except in the N-team. In a closed shop (as in the 1960's) professionals sat closely together and could share ideas and software. The Software Productivity, Quality and Reliability (SPQR) N-team is a good place for professional sharing.

A2) The Information Systems Division has a business thrust in Software Engineering and it is focused in the SEAL, the LaRC Center of Excellence in software engineering encouraged and supported by the Code Q Software Engineering Program. (The GSFC Software Engineering Laboratory just won the first IEEE Software Process Award; JSC has a Software Technology Branch; JPL has the SORCE).

A3) Perhaps the Center should form a local SPIN (software process improvement network) or SEPG (software engineering process group) in addition to the N-team.

Q) More than half of the software is being developed by people who are not software professionals. Engineers, doctors, and lawyers often write their own code. I can't find good textbooks written by professionals. Do you share that view?

A1) Perhaps we can never get non-software professionals away from programming. Would it help to have more training in software engineering?

A2) The Information Super-Highway may be more of a threat to a disciplined approach than interactive programming!

A3) The SEI apparently is now hiring mathematicians rather than computer science graduates, going back to the basics.

A4) I would defend the engineer who writes his own code. We need better practices in software development so the researcher can do the software work.

A5) Everyone needs to work more closely with the customer. The research engineer and the programmer need to work closely together.

A6) We need to have more fundamental training for "FORTRAN-type" programmers (basic training for research and prototype software).

Unfortunately, the training office doesn't like to repeat classes, which makes it difficult to offer basic classes to a wide audience.

Q) I build "Flight Systems" and there is electrical hardware that is not well-documented. Are we confusing software engineering with engineering as a discipline?

A) There is a difference between scientific research (prototyping) and systematic engineering (final product) software. Software engineering professionals should be involved with the final product.

COMMENT) We need to distinguish between scientific research and engineering development. Be careful not to compartmentalize or constrain research. I did the software development on one of my own mathematical models - it helped me to understand the problem.

In closing, Moderator Susan Voigt proposed 5 domains for software classification: flight software, facility software, ground support equipment software, management information systems, and research software (see Attachment). Also the intended use of software may affect its level of disciplined development: My use only, use within my work group at LaRC, Informal release outside LaRC, Beta release outside LaRC, and Formal release (e.g. COSMIC) outside LaRC.

Members of the audience were invited to comment on the domains and intended use categories and to join the LaRC software N-team if they were interested.

Perspective on Software Development

Charles E. Niles

What types of software do you develop? The domain is ground facility automation systems, specifically closed circuit and blowdown wind tunnels and research labs. Applications include control algorithms for test environment conditions (Mach number, Reynolds number, pressure, temperature), model support systems and other test articles (pitch, roll, yaw, Alpha, Beta), and high pressure air systems (pressure, temperature); process monitoring; operator interfaces; and utility functions such as data logging and sequence of events recording. Hardware systems include 80486-based microcontrollers, industrial PCS, PLCs, minicomputers, and combinations of these. Supporting systems include commercial analog and digital controllers, motion controllers, and servocontrollers.

Who are the users of this software? Facility operators, in support of LaRC and commercial aerospace researchers.

What is the life cycle (how long is the software used)? Indefinitely. Generally, the software is replaced during a CoF upgrade to computer hardware and control rooms.

Is there much maintenance or enhancement required? Steady work - correcting bugs and improving performance.

Maturity of Software Development - Software development, as we know it, has been around for 40-50 years. Software engineering, has been around since the early 1980's. Considering that other engineering disciplines have existed for 1900 years or so, the software world has come a long way. Understand though, that software engineering is still in its infancy. The point is that other engineering disciplines are not exact sciences and neither is software engineering. Only the laws of physics and the mathematics upon which they are based are.

Software Development Relative to Operating Systems - In recent years, more popular languages, notably C and C++, have advanced the portability of applications from mainframes, to minicomputers, to PCs, Macs, and workstations. Witness that different operating system platforms are capable of running the same application. However, the class of applications is generally confined to office automation tools. I believe that application software developers should be able to develop an application with no concern for the operating system it will run on. Of course, we have no universal operating system today and probably never will. But, the proliferation of operating systems and programming languages demands a consistent application programming interface. Currently, we have at least as many APIs as there are operating systems and hardware platforms to run them on. Perhaps, a universal API will emerge as the POSIX standards are developed.

Software Reusability - DoD mandated the use of Ada to promote reusability of source code, among other things. C++ was developed to foster the development of reusable class libraries and methods. Neither has accomplished this goal, and never will. **Problem - reusability is more trouble than it is worth.** How many of you have ever written a five-to-ten line routine to do something because there was not a library call to do it or because you could not afford the time to locate one ? ... How many of you have ever obtained source code that seemed to meet your needs but would not compile initially or did not execute as you expected ?... Individuals and small teams will usually reuse source code they have written themselves because they know where to find it and they know how it works - and it does not matter the language in which it is written - they will convert it, if necessary. But seldom will you go to another organization to find code that you need. Can you imagine Microsoft and Borland sharing source code ? Forget it. There are instances where commercial developers license software packages from other developers - it is less expensive than litigation. **Problem - new or unique software does not already exist.** An entirely new application can be based on a existing components (system calls, intrinsic functions, internally reused segments, etc), but they must be blended into a new overall package. Blending it all together to create a new application is still time-consuming, even if 50% of it is built from existing components... In my opinion, widespread software reuse will not happen on a nationwide basis and definitely not on an industry-wide basis. It is unlikely to be harnessed on a domain basis.

What is wrong with software developed at LaRC? What should we do about it?

1. Funding is always inadequate because of the politics involved in selling a facility modification project. The the higher the cost of a project, the less likely that it will be approved by HQ. When it is approved, the budget has been decreased too much to accomplish the overall job, let alone the software part. So, ultimately, we must complete the project in-house. The transition time for a 50,000-line job is not instantaneous. When we do a job in-house from its inception, the product is better, but the project takes longer because only minimal resources can be applied. The solution begins with properly planning a job and estimating the cost, including risk factors, and selling it for what it will cost, not for what management believes HQ will approve it.
2. Documentation is generally poor - it is usually outdated and incomplete. Face it, programmers like to write code, not documents. Programmers are extremely optimistic estimators. When they have used their allocated time getting their code to run, they do not have enough time to document anyway.
3. Management/customers do not understand the true costs of software. Most managers believe that software is something that comes on a set of disks or CD-ROM, costs \$500, and has a life of 6-12 months, depending on when the latest revision is released. Management fails to recognize that the developer probably spent \$1 million and 20 person-years to develop the initial release and must sell 200,000 copies to break even. Facilities automation personnel and/or contractor personnel, by comparison, have performed miracles with \$200,000 and 4 person-years. Unfortunately, our products have been overshadowed by late delivery and hardware reliability problems.

How can software improvements be institutionalized at LaRC?

1. Define standards and the criteria for their applicability.
2. Train and equip developers better.
3. Apply newer, industry-proven techniques.

Aside: Software has improved. Consider that the applications we develop today are significantly larger and more complex than their predecessors. I suspect that most of us could rewrite some piece of software we developed a few years ago in less time and far more robustly. So, what was it that really improved ?

What data should be collected on software developed at LaRC and how should this data be used?

1. Description of the software - function, size, platform, language(s), etc.
2. Resources applied - personnel, cost, tools
3. Why it was developed - benefits
4. Techniques - requirements analysis, design, coding, testing
5. Lessons learned

Information of this type could serve two purposes. First, a project team could use it for guidance. Secondly, after some period of time, a committee could evaluate this database to establish recommended practices, identify common attributes across different domains, identify common problems and how to avoid them, develop cost/resource criteria for future software projects, etc.

How can we encourage problem (defect) reporting & collection at LaRC?

There are two distinct categories: pre-release and post-release. Pre-release is the responsibility of the person(s) testing the software. Since the programmer usually performs initial testing, any data is virtually meaningless. Post-release is the responsibility of the users. I have found that encouragement is generally not an issue in this case. Problems having potential safety impact - a portion of which are software related - are reported when a subsystem fails. Under the facilities Configuration Management program, the Facility Safety Head is responsible for reporting such problems. Problems are generally reported when a certain function of the software becomes important to a test. Generally, there is no mechanism to report problems specifically with software at LaRC.

What suggestions would you make for how we should be developing software at LaRC in the future? I believe an individual representing each software domain (i.e. Blue team) should visit the SEI or a major commercial developer, spend a few days observing, return, draft a software development handbook, obtain feedback from a different set of individuals representing each software domain (i.e. Red team), revise the handbook, publish it, and encourage management to enforce it.

Perspectives on Software Development

**Thomas A. Zang
Chair, LCUC
Head, MDO Branch, RTG**

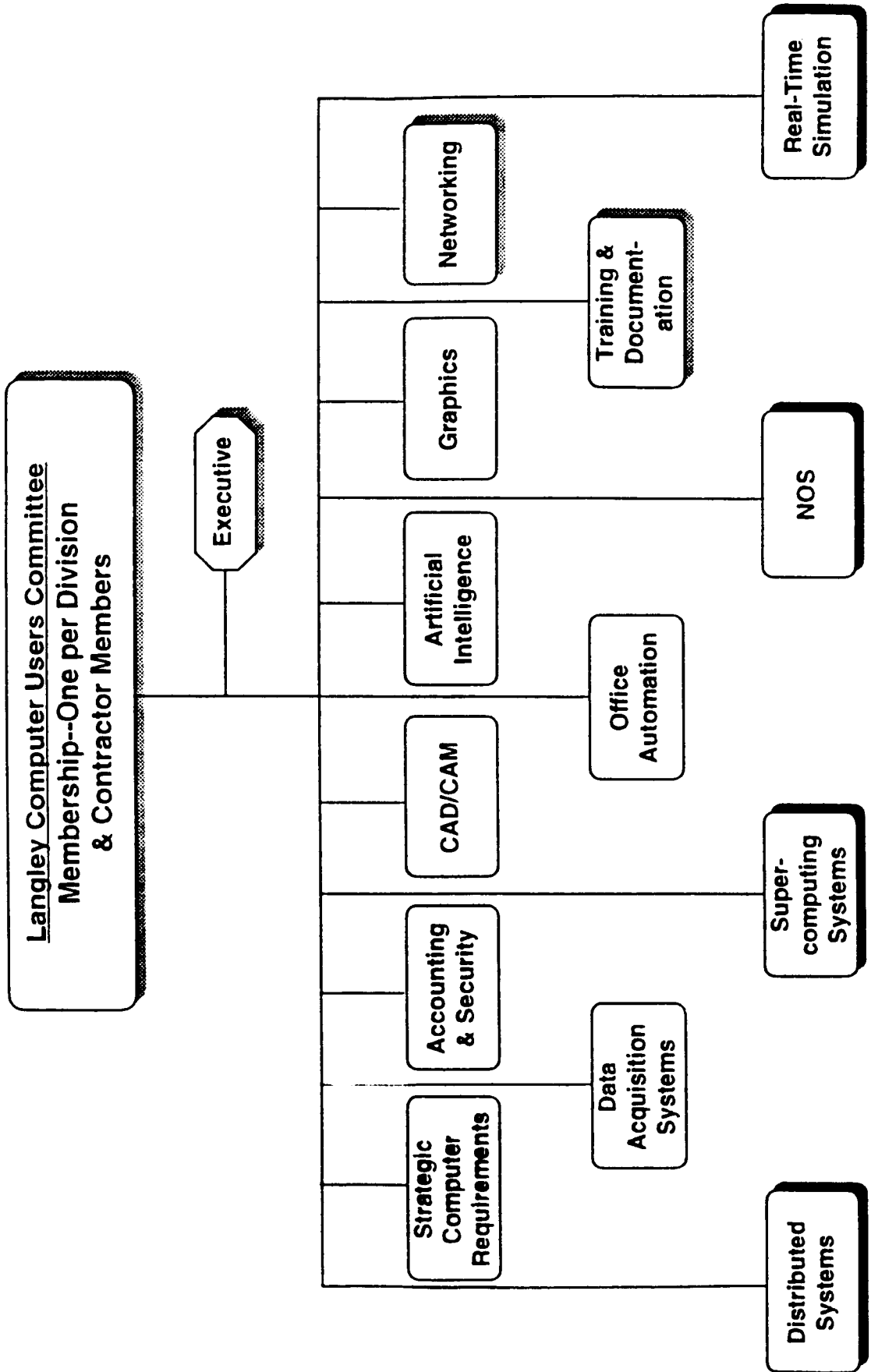
**CSTC Workshop
June 15, 1994**

Langley Computer Users Committee

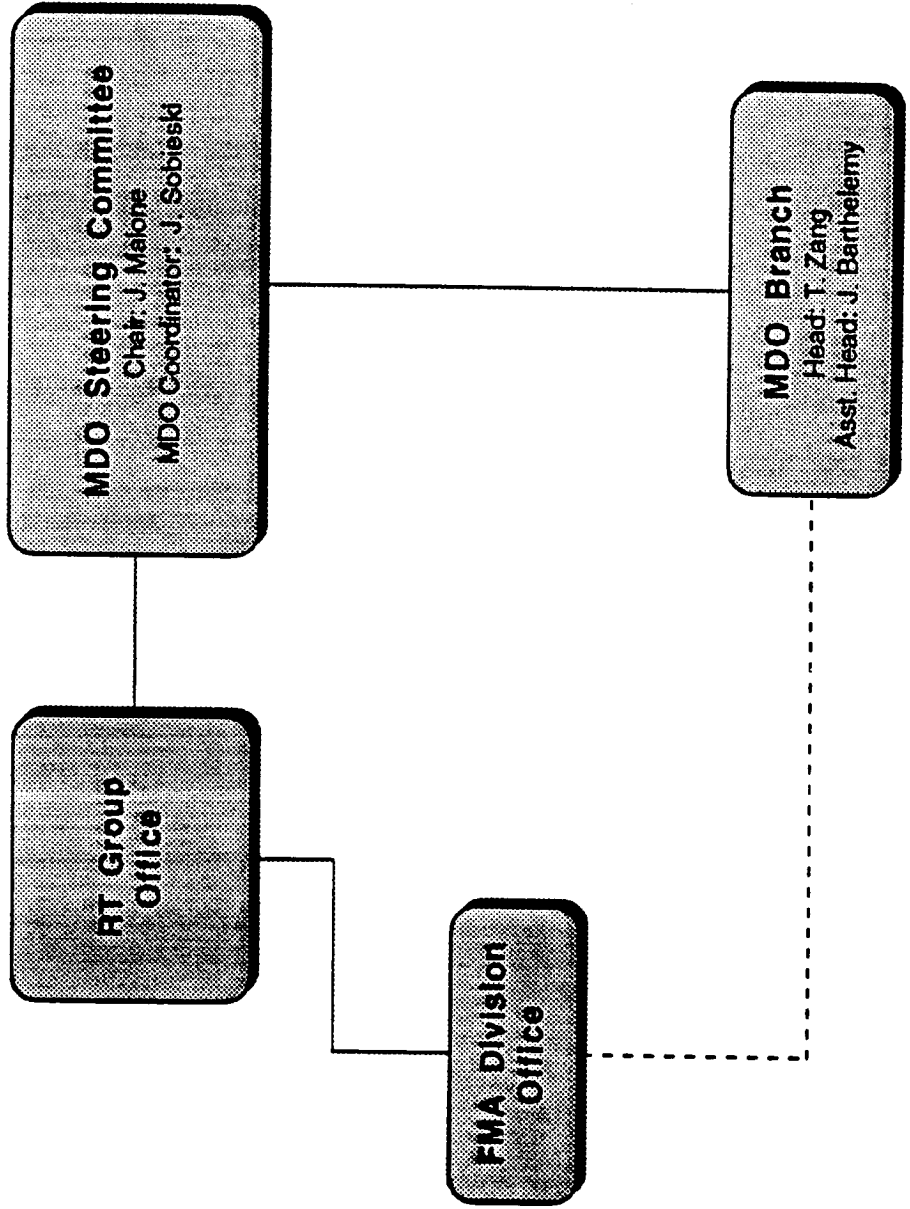
- **Membership:** research & support staff
- **Role:** a grass roots users organization that provides
 - tactical recommendations (solicited & unsolicited) to providers on computing issues
 - provides forums for information exchange between users on a variety of computing topics
- **Audience:**
 - providers (ISD)
 - computer users
- **Will be reorganized in late Summer**

Langley Computer Users Committee

Existing Structure



RTG MDO Organization



MDO Branch Functions

- **Perform research into the methodology of MDO**
- **Participate in MDO application studies**
- **Foster the growth of multidisciplinary activities throughout the RTG**
- **Transfer new MDO methodologies to RTG, APG, SASPG, industry**

NASA Products

- **Concepts**
 - NASA disseminates the *concepts* underlying new *algorithms*, *physical models* and/or *computer implementations*
 - these *concepts* are incorporated into customer's own computer codes
- **Portable Modules**
 - these modules are incorporated into customer's existing codes to add or improve capabilities
- **Pilot Codes**
 - early release of a research code that is typically special purpose, requires an expert user and is not fully debugged
 - customer uses full code or else extracts modules of interest
- **Production Codes**
 - robust, efficient, user-friendly, validated, supported
 - customer uses full code or else extracts modules of interest

**A Proposal: How to Improve NASA-
Developed Computer Programs**

by

Jaroslav Sobieski

**Proposed for NASA Software Technology
Workshop, March, 1980**

(Presented to LCUC, 1980)

Sobieski, 1980

Problem

- **A large number of computer programs are generated in NASA research work as prototype, proof-of-the-concept programs, typically as companion items to NASA formal reports**
- **These programs are, typically, researcher-generated and frequently embody new important methodologies and solutions**
- **Problem is that too often these programs are unreliable and/or inefficient, especially in hands of users other than developers**

Evidence

- A particular group of NASA and contractor engineers (total of 14) have been working for the past 6 years at NASA LRC with an integrated system of programs applied in aircraft aerodynamics, aeroelasticity and structures problems
- The system involves now 43 programs ranging in size from 500 to 30,000 Fortran lines, a statistically significant sample
- Majority of these programs came from NASA research work, in-house and grant
- Nearly all of these programs did not work as claimed when they were first obtained, significant time and effort were needed to make them work and to make them operate with a minimum of acceptable efficiency
- In contrast, most of the contractor generated programs were free of the serious reliability and efficiency problems

Diagnosis of the Problem

- In a typical twin product of a research work, formal report plus computer program, report will receive all the attention of the organization, program will receive none
- The research organization is geared to support quality report production (editorial committees, illustrators, reproduction, etc.) but not quality program production
- Impression is created that in the report-program pair, the program's status is secondary. Consequence: no incentive to strive for program quality
- Vast difference in the type of skills and degree of effort between a prototype, proof-of-the-concept program, and a professionally coded and tested and documented program
- A researcher is a natural creator of the former but is ill-equipped to develop the latter

Reflections from a “Jurassic Programmer” on Software Development at LaRC

In 1972 the software development environment at NASA was very different. It was a batch environment where the programmer’s life revolved around the deliveries of the “green tub” and the survival of data and programs on assorted paper media. Some advanced programmers took advantage of 7-track tapes and data cells for storage.

- *The Revolution of 1975*

Two major developments occurred at the Center in 1975 that changed the scope and way of doing software development forever.

The advent of micro-processors ended the monopoly held by discrete hardware components or “random logic” in the implementation of control functions. This also exposed engineers to “programmers” who were unfamiliar with hardware and its associated engineering discipline. Critical real-time applications were now in the hands of software developers and opened up the embedded domain. Good programmers saw the value in adopting practices very analogous to those of the hardware designers. As the electronics revolution continued, hardware engineers were forced to become somewhat familiar with software.

The introduction of the interactive development environment was brought about by the installation of a new operating system on the NOS mainframes and the populating of selected offices with dumb terminals. Key-to-disk storage did away with all of those card files. The terminal opened access to any programmer, regardless of background. Requirements to pass proficiency tests on the use of the system and FORTRAN in order to get a user number were deleted.

- Q&A: *Was batch all bad?*
 Was interactive all good?

The answers to the two questions are No and No. In retrospect, I believe the batch environment had several good attributes, and the interactive environment has been a major factor in the lack of discipline we see today.

Pre-revolutionary programmers realized the value of desk checking and flowcharting because it could take weeks to get a successful compilation if they weren’t careful. Plotting in a batch mode was often an extremely frustrating task! Programmers were freed from the tedium of keypunching because folks at ACD punched and verified from green and white coding sheets. This gave me an opportunity to insert lots of good documentation and scan the code one more time before committing to the initial submittal. The slower pace of life gave programmers more time to sit and stare at their code. In fact, managers expected them to behave this way.

In the interactive world, the lure of instant gratification at the terminal led to a rush to the CRT. People routinely sat down and began typing wildly without even a coding sheet. The most unfortunate result was that people could more easily confuse activity with productivity. Often, the lowest level task - pounding the keys - was the key measure of productivity.

- *The Revolution of 1994 - better, cheaper, faster?*

Here at Langley, times have changed. In fact, times are tough. Software is now a real product, not just a by-product generated along the way to some higher goal like a report. Software is a technology that needs to be transferred outside the gate - and it needs to be good because of its added visibility. Quality issues are brought up everywhere. The dilemma is how to get quality while operating under a constrained budget.

- *No more heroes - we have to work smarter*

There is no more of the "green medicine" to throw at our software problems. There are no additional people to hire. We must realize that faster CPUs and graphics workstations and glitzy tools simply speed up the most visible portion of the development process. Automating a poor process will get us nowhere.

We need to create a recipe for successful software development for the various domains at LaRC. That is, learn from the mistakes that are often the best teachers, share the tips and tricks, and reward the people who do the right things throughout the entire lifecycle that result in quality software. We need to catch our collective breath and treat software like an engineering discipline in order to design, manage, document, maintain, and transfer knowledge.

In short, there is no license to meander anymore. The choice is ours: will we remember the past or are we, as Santayana says, "doomed to repeat it"?

Pamela L. Rinsland

LaRC Software Domains

- **Flight software**
Software that performs command, control, onboard data processing, data storage and communication for space (e.g. LITE, HALOE) or aircraft (e.g. LASE, CLASS, Windshear) instruments.
- **Facility software**
Software that performs the command, control, data acquisition for key LaRC facilities such as tunnels, flight simulators, hangar data systems, experimental aircraft infrastructure (737, 757), and other test facilities.
- **Ground Support Equipment**
Software used to perform test and integration, check out flight instruments, and monitor system performance during mission operations.
- **Management Information Systems**
Personnel and financial resource management software used to support LaRC resource management.
- **Research Software**
Engineering analysis tools, simulations and models developed to support the research mission of the Center such as CFD codes, data presentation and visualization, models & algorithms, and post-mission data analyses.

Software Engineering & Ada Lab (SEAL) Tools

- 1) CADRE Teamwork CASE Tools:
 - a) Teamwork/SA (Structured Analysis)
 - b) Teamwork/RT (Real-Time Analysis)
 - c) Teamwork/IM (Information Modeling)
 - d) Teamwork/SD (Structure Design)
 - e) Teamwork/OOD (Object-Oriented Design)
 - f) Teamwork/Ada (Editor, Code Generator, Design Sensitive Editor)
 - g) Teamwork/SIM (Simulation Tool)
 - h) Teamwork/FORTRAN REV (rev. eng.)
 - i) Ensemble "C" Tools
 - System Understanding (High-level rev. eng.)
 - Function Understanding (Low-level rev. eng.)
 - Documentation

- 2) Paradigm Plus (Object-Oriented Meta-CASE Tool) (4)

- 3) McCabe Tools:
 - a) Analysis of Complexity Tool (ACT)
 - b) Battlemap Analysis Tool (BAT)
 - c) Ada language parser

- 4) Ada Measurement and Analysis Tool/Diana (AdaMAT/D)
- 5) VAX Software Engineering Tools (VAXset)
- 6) NASA Intelligent Documentation Management System (IDMS)
- 7) InQuisiX - Reuse Repository

- 8) In-Circuit Emulators
 - a) Microtek MICE-V 386 Emulator
 - b) Microtek MICE-V 486 Emulator
 - c) HyperSource-386/486 Source/Assembly-Level Debugger
 - d) AMC ES-1800 80186 Emulator (2)
 - e) Emulation Support Driver (ESD) Software

- 9) CADRE Software Analysis Workstation (SAW) (2)
 - a) Interactive State Analyzer
 - b) SoftAnalyst
 - c) Probes for 80186/286/386, 1750A, Generic

- 10) Logic Analyzers/Oscilloscopes:
 - a) HP 16500A Logic Analyzer
 - b) HP 16530A Digitizing Oscilloscope Module
 - c) HP Probes/Preprocessor Interfaces for: 1553B, TMS320C30/31, 80486, HP-IB-RS232-RS449, SCSI Bus, user definable
 - d) HP Performance Analyzer
 - e) HP Inverse Assemblers
 - f) Fluke Scopemeters (2)

- 11) TITAN SESCO Flight Equivalent Computer
 - a) SECS 386/30 Single Board Computer
 - b) SECS 186/30 Single Board Computer
 - c) SECS 80/1553B Single Board Computer
 - d) Memory board (386 - 4M, 186 - 512K)
 - e) Parallel and Analog I/O Modules

- 12) PROM Tools
 - a) TITAN/Data IO Flight Board Programmer
 - b) EPROM Erasers (3)
 - c) PROM ICE

- 13) PC Data Acquisition Hardware and Software
 - a) GPIB Boards and Software
 - b) AT-DIO-32F (10) AND DIO-96 Boards and software
 - c) SF-1 (2) Shuttle SFMDM Cards
 - d) LabVIEW For Windows Dev. System (2)
 - e) LabWindows
 - f) NI-DAQ DOS/Windows

- 14) Systems
 - a) VAXstation 4000 model 60
 - b) SUN SPARCstation 10
 - c) SUNserver 690MP
 - d) Novell 486 Server/UPS
 - e) Castelle FAXpress
 - f) SMTP Gateway PC
 - g) Various 386/486 PCs
 - h) Laser Printers (3)

- 15) Miscellaneous
 - a) Soldering/Desoldering station
 - b) Wire-wrap tools
 - c) Insertion/Deinsertion tools
 - d) Proto-Boards/Breadboards
 - e) Military & D-shell connectors and cabling tools
 - f) HP Power Supplies (4)
 - g) Optical Drives

For more information, contact Jerry Garcia at (804)-864-5888.

SESSION 3 Software Engineering Standards, Methods, and CASE Tools

Chaired by

Susan Voigt

- 3.1 Model-based Software Process Improvement - Brenda Zettervall**
- 3.2 A Study of Software Standards Used in the Avionics Industry - Kelly Hayhurst**
- 3.3 A Software Tool for Dataflow Graph Scheduling - Robert Jones**
- 3.4 Use of Software Through Pictures on CERES - Troy Anselmo**

Model-Based Software Process Improvement

P

Brenda T. Zettervall
Naval Surface Warfare Center (NSWC)
Port Hueneme Division (PHD)
East Coast Operation (ECO)
Dam Neck
Virginia Beach VA

This presentation demonstrates our organization's approach to model-based Software Process Improvement (SPI). Our organization, a Process Transfer Technology Affiliate of the STARS program, was selected in April 1993 to participate as a field test site for the Software Engineering Institute (SEI) Software Process Definition (SPD) project. The products tested included the improvement model itself, descriptive modeling techniques, the CMM level 2 framework document, and the use of process definition guidelines and templates.

The SPI model developed by the SPD project at the SEI represents a five stage cyclic approach for organizational process improvement. The cycle consists of the initiating, diagnosing, establishing, acting, and leveraging phases. Our organization's three year Total Quality Initiative facilitated the adoption of this model for our software improvement teams.

The process improvement infrastructure includes the steering committee, SEPG team leader, the SEPG core advisors, Quality Management Boards (QMB), and designated working groups chartered by the SEPG. The QMB's directly support the strategic goals of the organization. Monthly briefings from the SEPG team leader to the steering committee and the QMB's facilitate the integration of the SPI initiative with the strategic business goals.

The SPD project at SEI field-tested the Process Framework Document for CMM level 2 at our organization. The document provides checklists to determine CMM compliance for each Key Process Area (KPA). In addition, we gained insight into the necessary organizational components to support well-defined processes.

Process Definition (PD) training was provided for our SEPG, Technology QMB, and the Project Planning Working Group. Our SEPG recognized the need to establish a documented standard approach for PD that all software improvement teams can use (i.e. a well-defined process!). Our Process Breakdown Structure establishes planning, definition, and enactment as the top-level phases of the Process Engineering life-cycle.

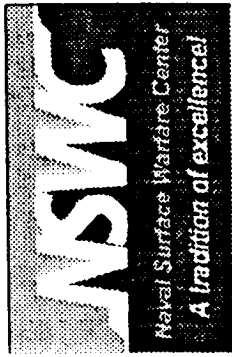
The process planning phase is necessary to baseline and document the current process by establishing the purpose and the high-level process flow. In addition, it is important to set the policy that will over-arch the process and help set the context for the follow-on process definition engineering. The process definition phase is decomposed into three activities: layout, design, and enactment information. The layout activity establishes the process relationships by organizing the high-level entry/tasks/validation/exit (ETVX) information and defining the work flow and work products associated with the process. In addition, a mid-level process flow is established during this step which will facilitate using the information organizers in the design activity. The agents that will perform each task are also identified during this activity.

The design activity of the definition phase is characterized by the use of multiple information organizers (i.e. templates) which provide the necessary data to develop the enactment information. Measurement criteria and the validation method are further defined in this stage of process definition.

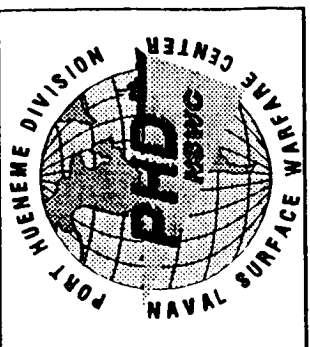
The development of the enactment information is the last activity to be performed in the definition phase of process engineering. The procedures must be developed during this activity in order to trial test the process during a pilot project. The training requirements for the process must also be established at this time.

The enactment and process support is the final phase of process engineering and constitutes the institutionalization of the process. This phase must establish process control and process assurance procedures to ensure that the process has the ability to improve. A training plan is important to support the on-going use of the process.

The outer ring represents all of the work products developed during the process engineering life-cycle. In an attempt to avoid shelfware, the SEPG is targeting a Process User's Manual for each KPA that will contain only the essential information required for the user of the process.



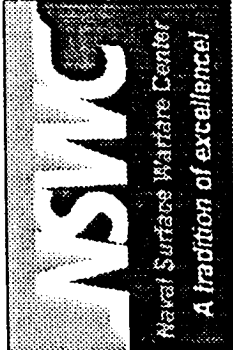
NAVAL SURFACE WARFARE CENTER
PORT HUENEME DIVISION
EAST COAST OPERATIONS



MODEL - BASED SOFTWARE PROCESS IMPROVEMENT

Ms. BRENDA T. ZETTERVALL
QUALITY IMPROVEMENT ADMINISTRATOR

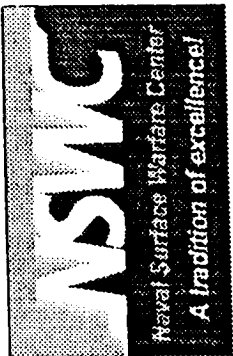
1920 REGULUS AVE
DAM NECK
VIRGINIA BEACH VA
23461-2097
BZETTERVALL@HERCULES.NSWSES.NAVY.MIL



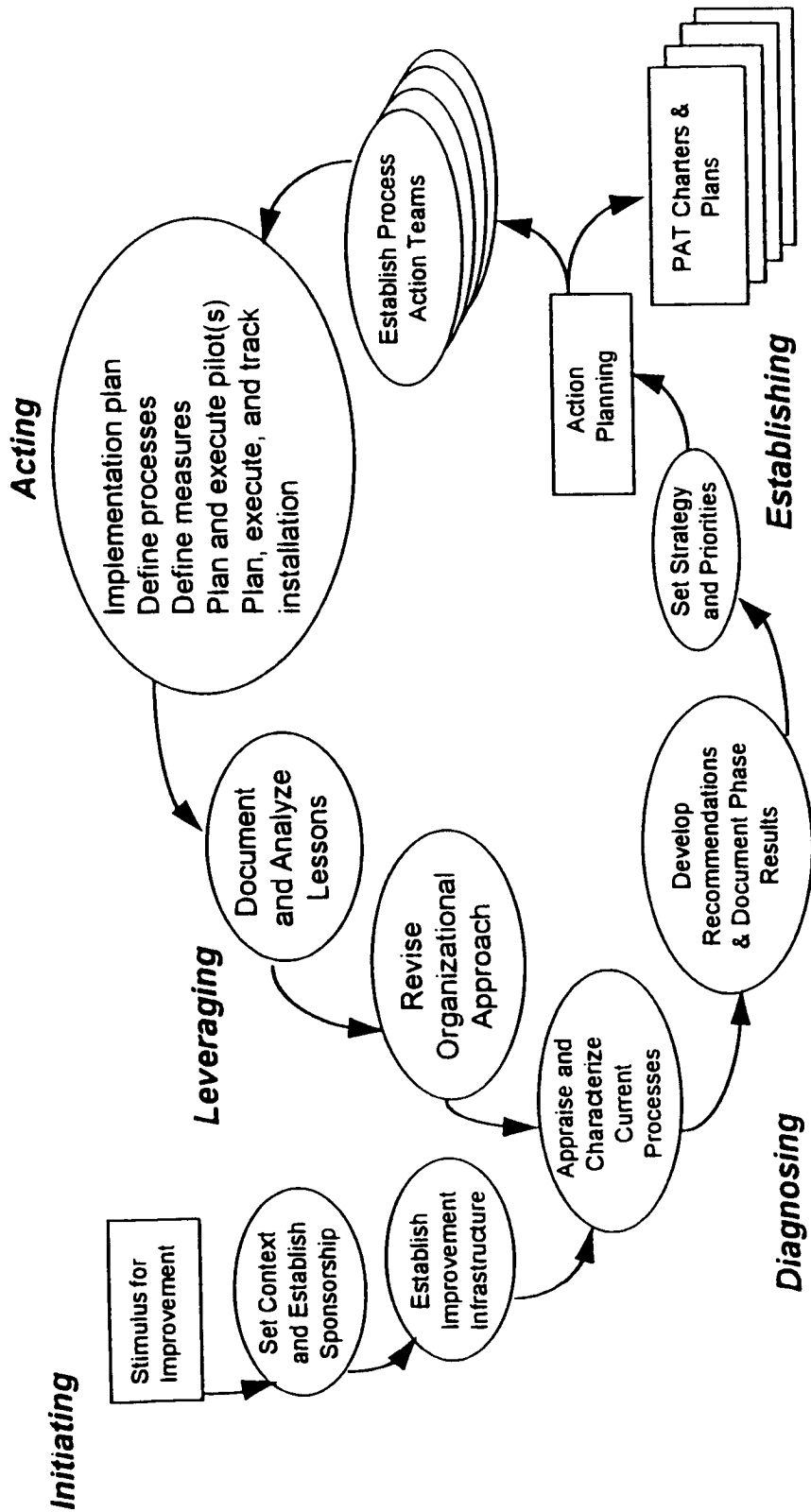
AGENDA

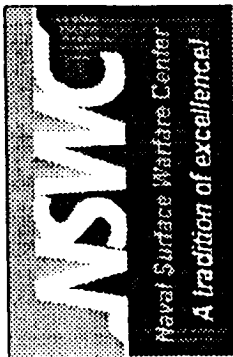


- SPI MODEL**
- IMPROVEMENT INFRASTRUCTURE**
- ESSENCE OF LEVEL 2**
- PROCESS DEFINITION FRAMEWORK**
 - PLANNING**
 - LAYOUT**
 - DESIGN**
 - ENACTMENT**
- PROCESS MANUAL**

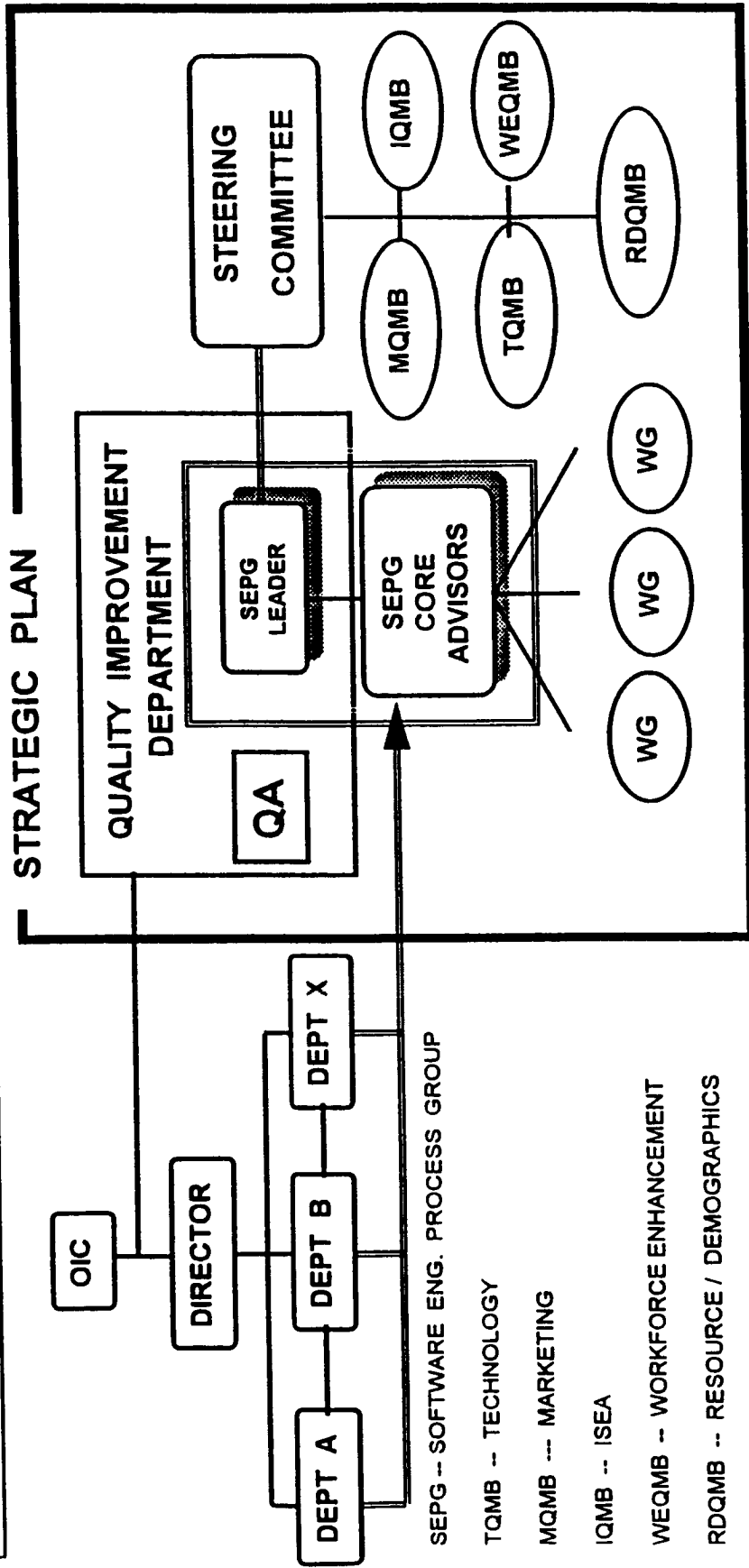


SOFTWARE PROCESS IMPROVEMENT MODEL

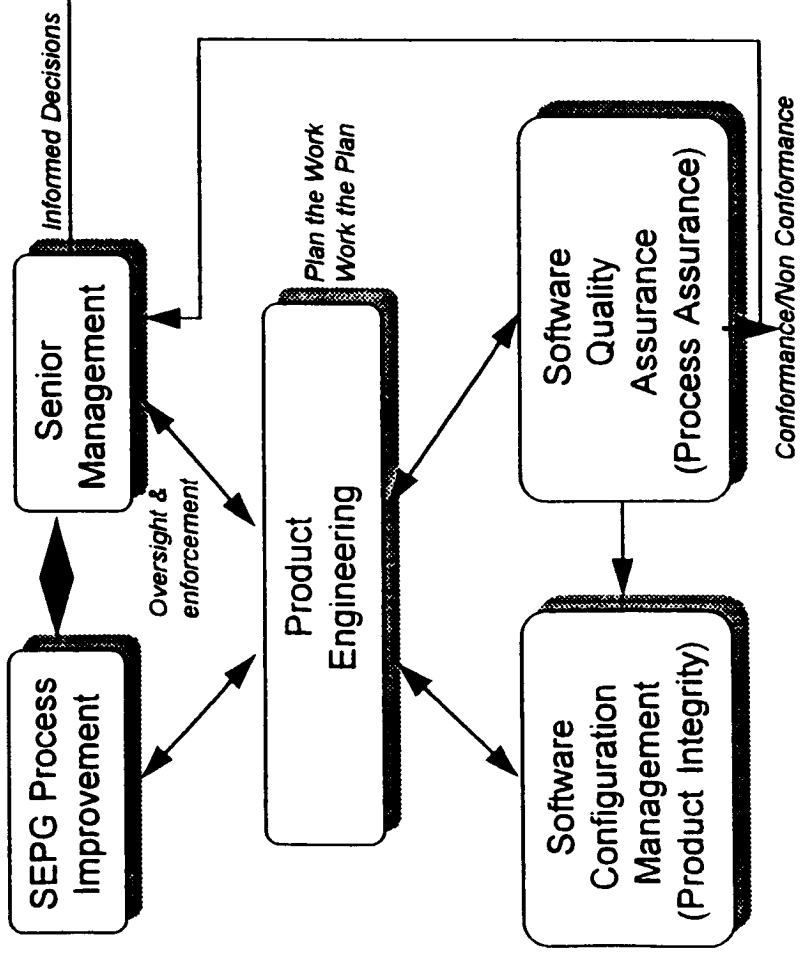
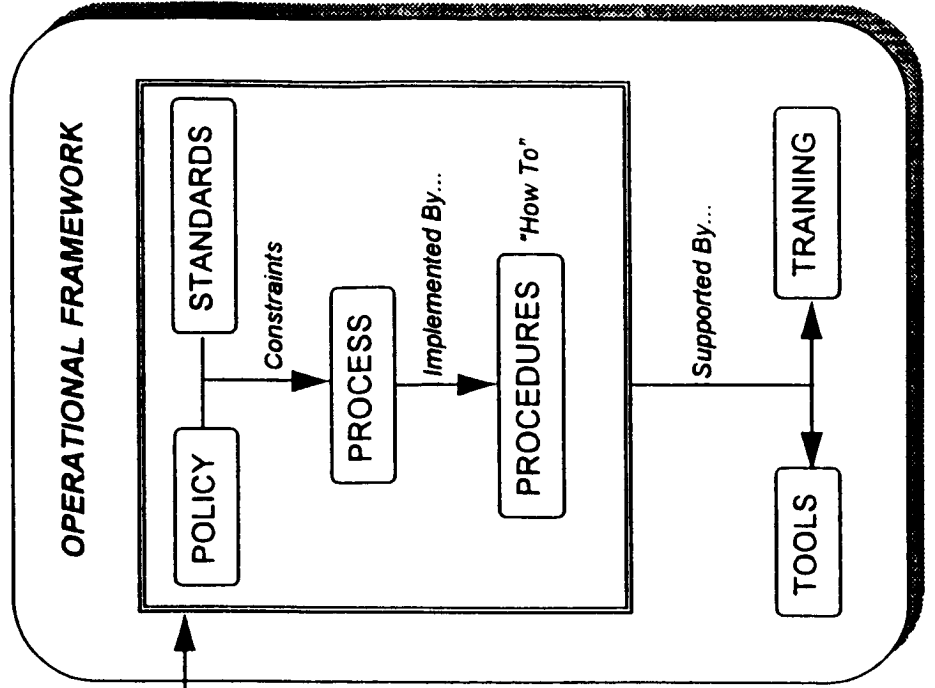





NSWC - PHD - ECO PROCESS IMPROVEMENT INFRASTRUCTURE




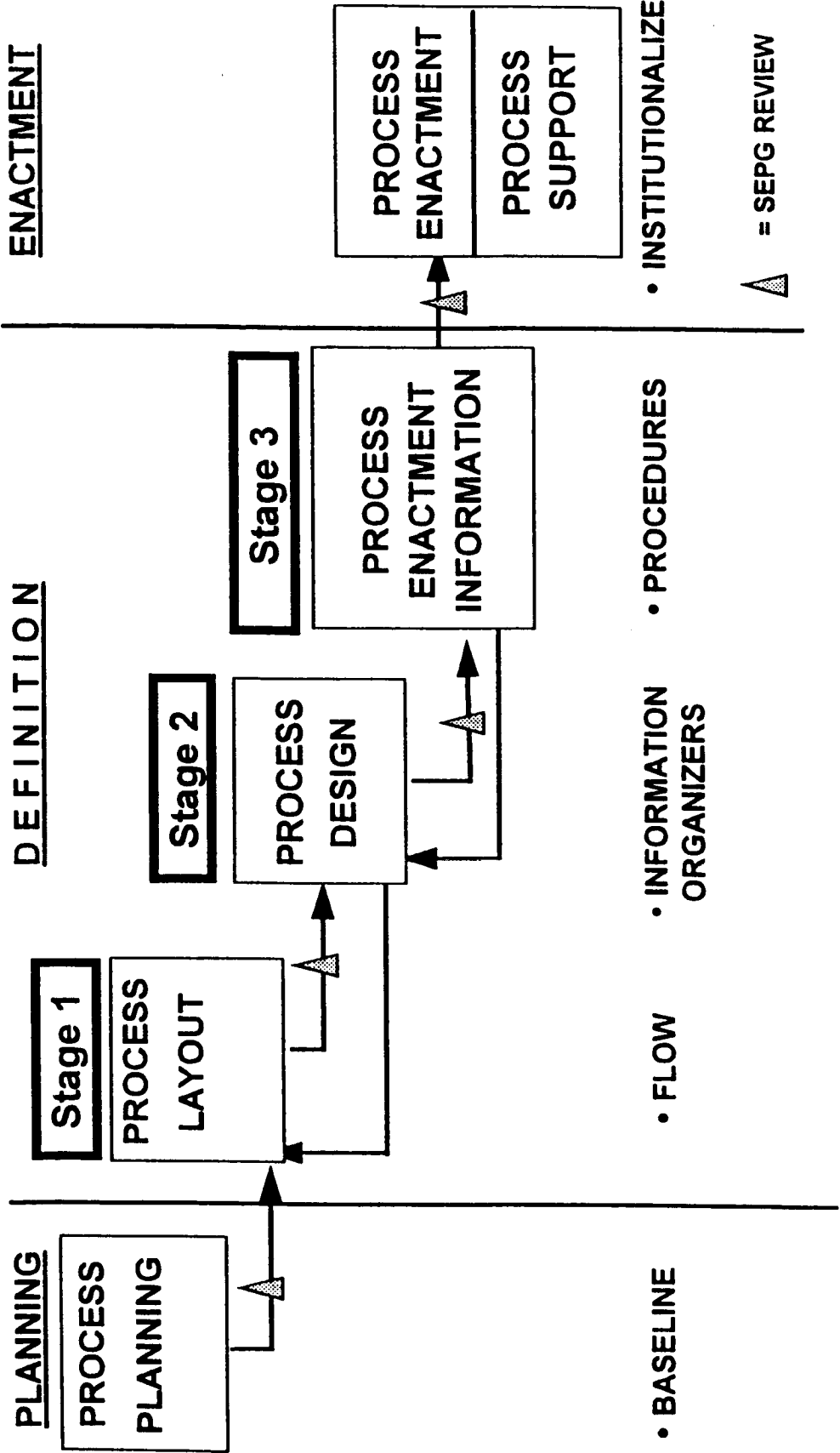
THE ESSENCE OF LEVEL 2 "TAKES COMMITMENT AND TIME !"





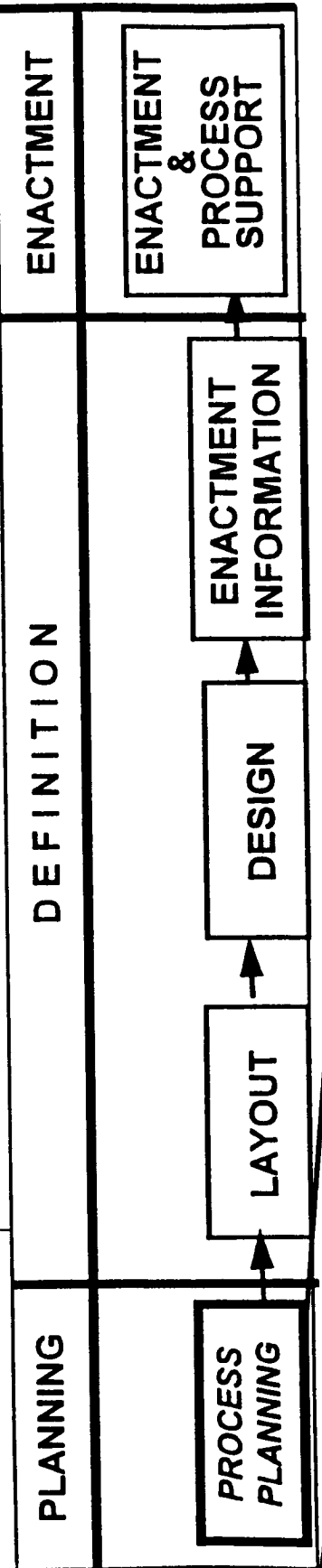
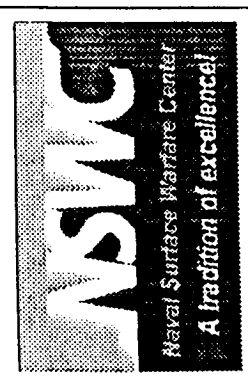
NSWC PROCESS DEFINITION FRAMEWORK







PROCESS DEFINITION FRAMEWORK



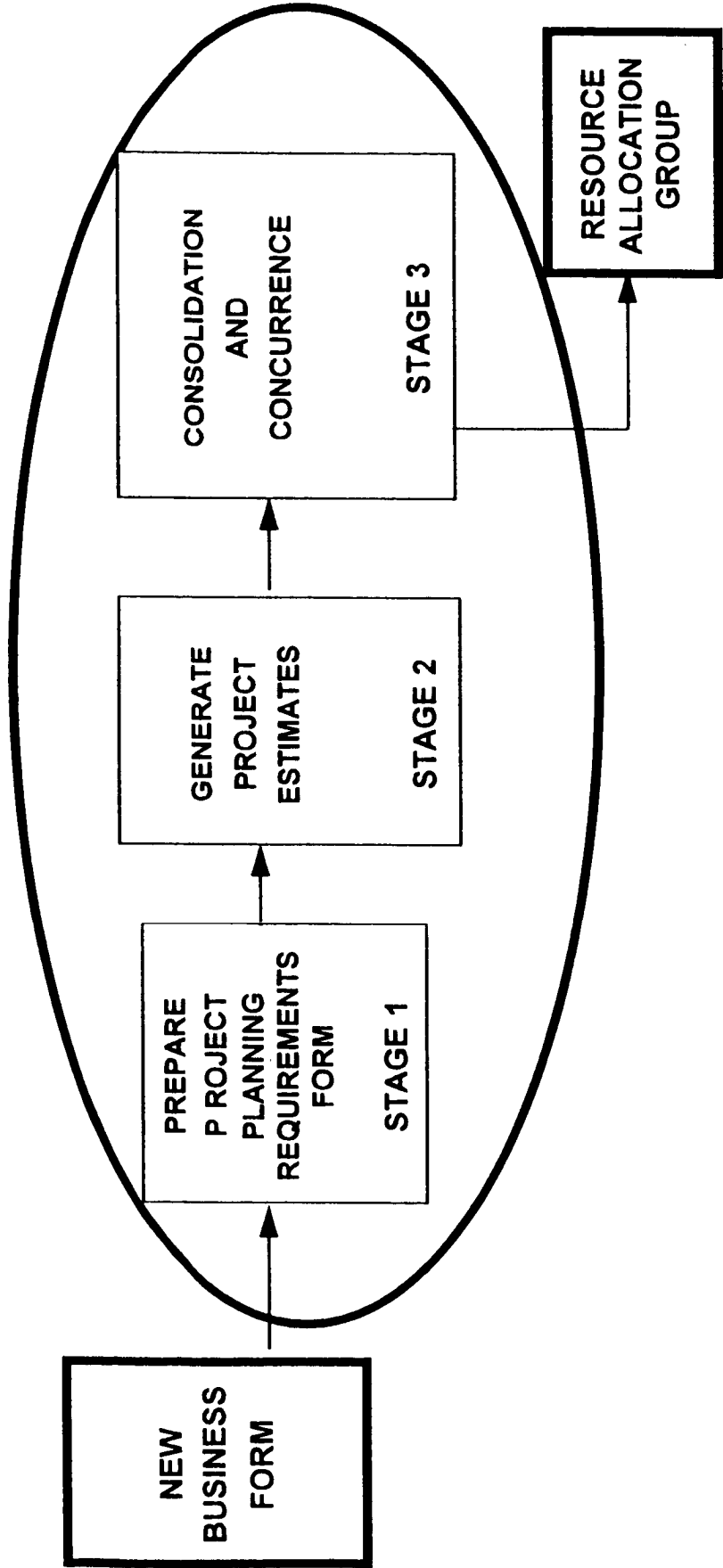
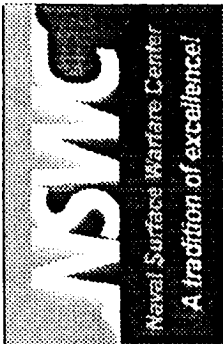
- BASELINE
 - DATA GATHERING
 - AUDIENCE
 - ROLES
 - USAGE
 - KNOWLEDGE
 - MOTIVATION
 - INTERVIEW

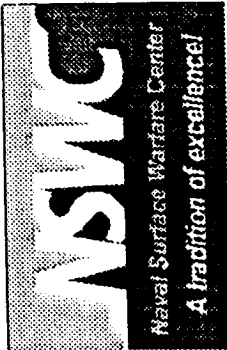
- HIGH LEVEL PROCESS FLOW

- POLICY STATEMENT



PROJECT PLANNING COMMITMENT PROCESS





PROCESS DEFINITION FRAMEWORK



PLANNING

PROCESS
PLANNING

DEFINITION

LAYOUT

DESIGN

ENACTMENT
INFORMATION

ENACTMENT

ENACTMENT
&
PROCESS
SUPPORT

- PROCESS RELATIONSHIPS
- INITIAL MODEL
-ETVX
- MID - LEVEL PROCESS FLOW
- WORK PRODUCTS DEFINED



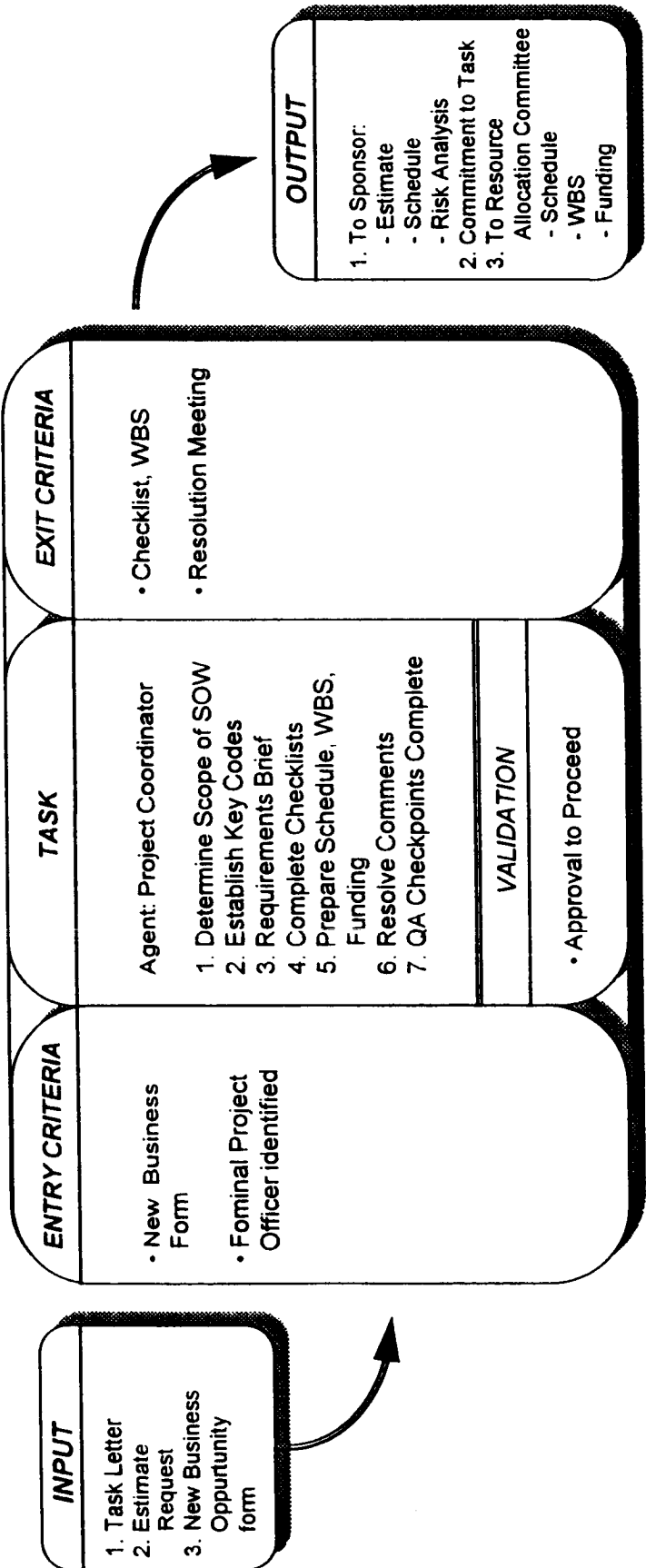
THE ETVX PROCESS

(ENTRY, TASK, VALIDATION, EXIT)



TASK: Project Planning (Bids, RFP, Potential)

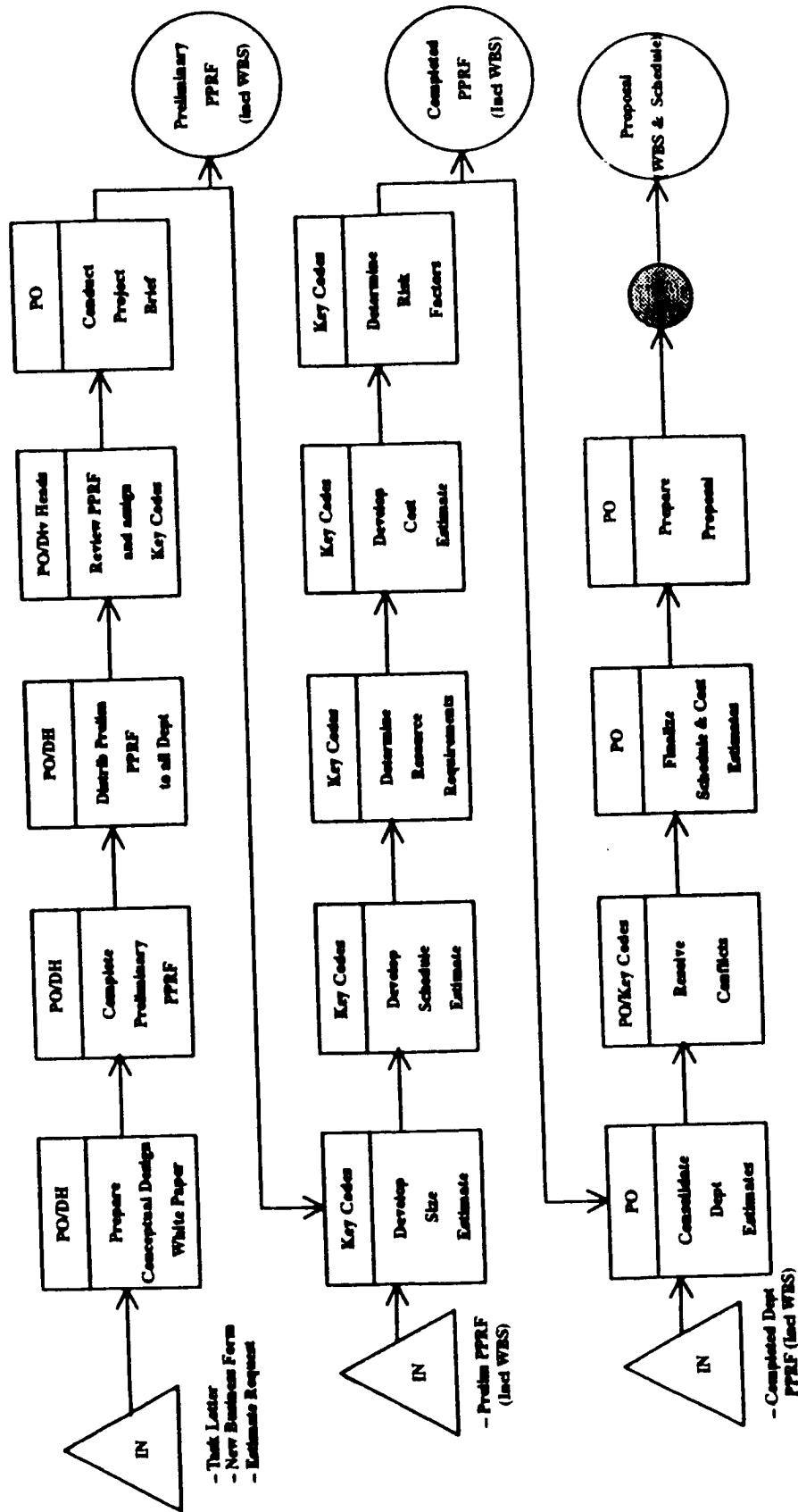
PURPOSE: Ensure all Departments and Divisions are involved in Project planning process



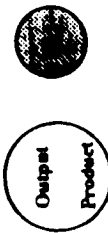
PARENT: Project Officer's Handbook

ACTION CODE: Project Coordinator

Project Planning Process (Bid, Proposal, Potential)



QA Review



Legend:



PO = Project Officer
 DH = Department Head
 PPRF = Project Planning Requirements Form



PROCESS DEFINITION FRAMEWORK



ENACTMENT

ENACTMENT
&
PROCESS
SUPPORT

DEFINITION

ENACTMENT
INFORMATION

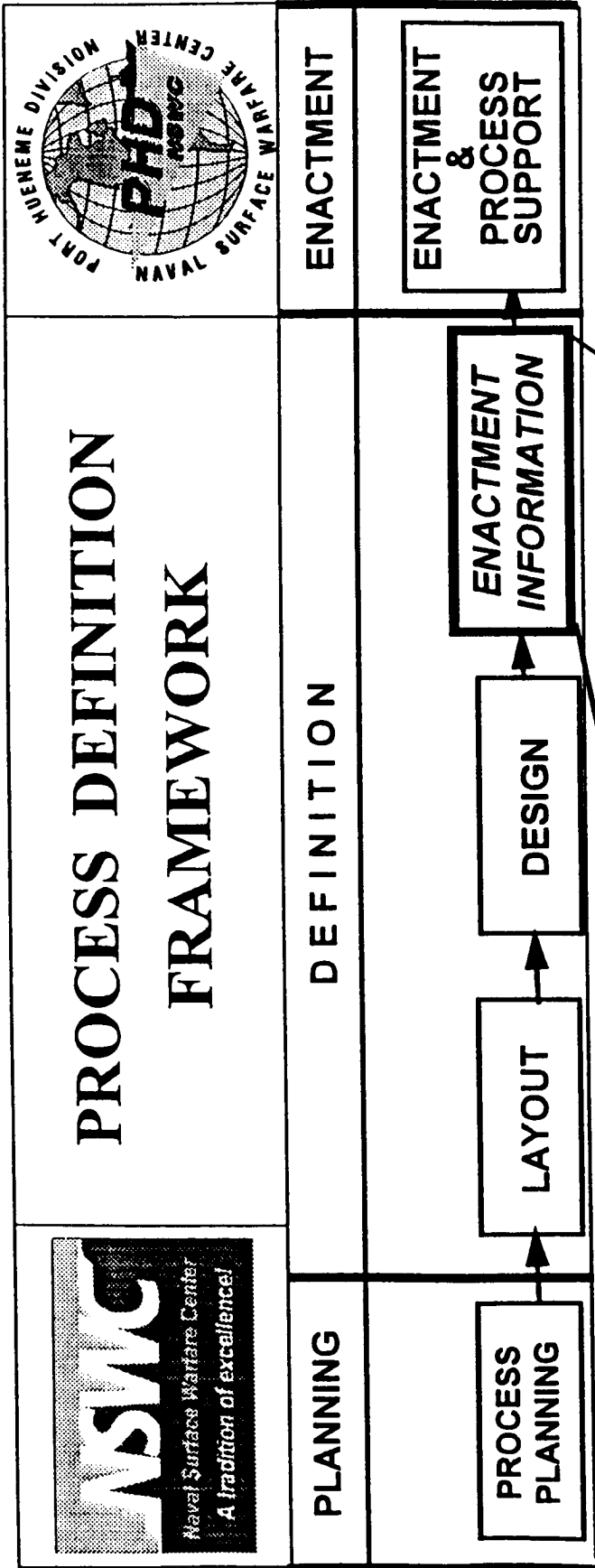
DESIGN

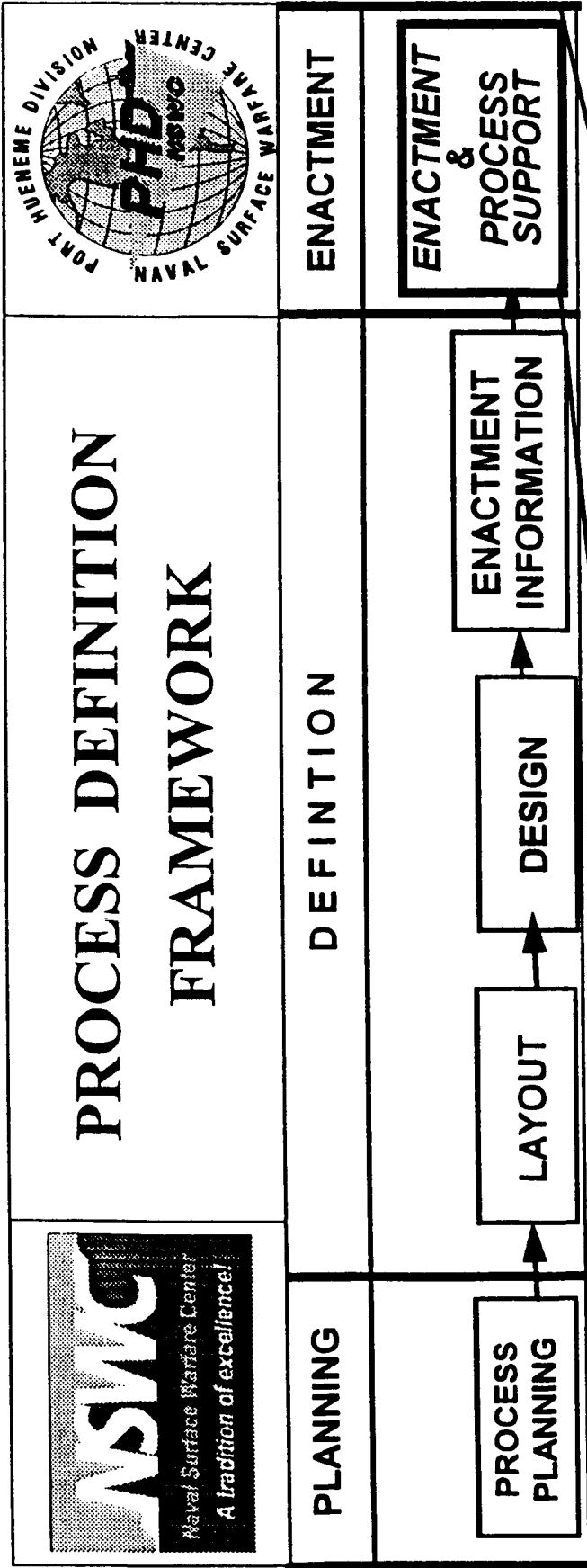
LAYOUT

PLANNING

PROCESS
PLANNING

- INFORMATION ORGANIZERS
 - INPUTS
 - OUTPUTS
 - TASKS
 - AGENTS
 - MEASUREMENTS
 - VALIDATION

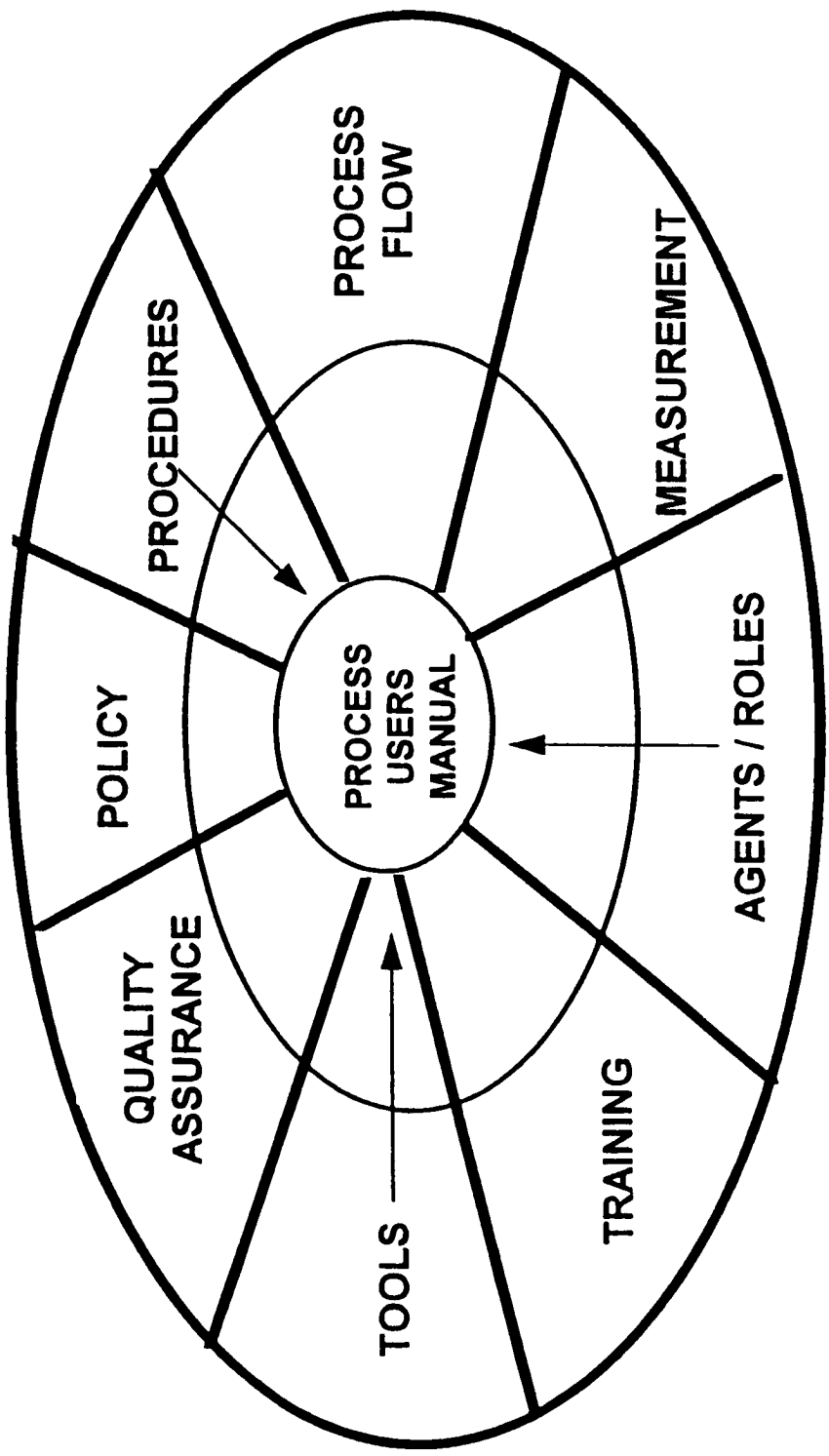




- INSTITUTIONALIZATION
- PROCESS CONTROL
 - MONITOR
 - MEASUREMENT
 - ANALYSIS
 - IMPROVEMENT
- PROCESS ASSURANCE
 - TAILORING / GUIDANCE
 - ENFORCEMENT
- TRAINING PLAN



KEY PROCESS AREA PROCESS MANUAL



A Study of Software Standards Used in the Avionics Industry

Kelly J. Hayhurst
Assessment Technology Branch
Research Technology Group

356042

P. 21

Within the past decade, software has become an increasingly common element in computing systems. In particular, the role of software used in the aerospace industry, especially in life- or safety-critical applications, is rapidly expanding. This intensifies the need to use effective techniques for achieving and verifying the reliability of avionics software. Although certain software development processes and techniques are mandated by government regulating agencies such as the Federal Aviation Administration (FAA) and the Department of Defense, no one methodology has been shown to consistently produce reliable software. The knowledge base for designing reliable software simply has not reached the maturity of its hardware counterpart.

To date, existing software development methods and standards have been accepted largely based on intuitive arguments or anecdotal evidence. The data typically collected from a software development process include a description and some classification of faults identified during the prescribed development and verification activities and the final software product. From a statistical perspective, this represents a single replicate of development information. From this single replicate, some insight could be gained into the feasibility and impact of the software development method on that particular implementation of software. However, the single replicate does not provide enough information to make statistical inferences with confidence about the effectiveness of the development method in general and it provides little information about the operational behavior of the software. To provide the empirical data necessary to scientifically evaluate and improve software processes and product reliability, controlled experimentation that accounts for the performance of software during operation is needed.

In an effort to increase our understanding of software, Langley Research Center has conducted a series of

experiments over the past 15 years with the goal of understanding why and how software fails. With an increased understanding of the failure behavior of software, improved methods for producing reliable software and assessing reliability can be developed. As part of this program, the effectiveness of current industry standards for the development of avionics software is being investigated. This study involves the generation of a controlled environment to conduct scientific experiments on software processes.

The Guidance and Control Software (GCS) project involves the establishment of an experimentation test-bed to monitor and study the application of software development methods and collect data that can be used to make statistical inferences about the effectiveness of those methods. This test-bed allows the development and simulated operational testing of multiple implementations of a guidance and control application that was adapted from the terminal descent phase of the Viking lander. The test-bed is comprised of software requirements for the guidance and control application, a configuration management and data collection system, and a software simulator to run the control software in a simulated operational environment. The simulator is designed to allow one or more implementations of the GCS to run in a multitasking environment and to collect data on the comparison of the results from multiple implementations.

This test-bed provides a capability for empirically investigating the effectiveness of software development methods along with investigating the reliability of the resultant software. Currently, the GCS test-bed is being used to investigate development and verification techniques that comply with the Requirements and Technical Concepts for Aviation RTCA/DO-178B guidelines, "Software Considerations in Airborne Systems and Equipment Certification." The DO-178B guidelines are used by every commercial civil transport airframer and equipment vendor since compliance with

these guidelines is required by the FAA for developing software to be used in systems or equipment certified for use in commercial aircraft.

The purpose of the DO-178B document is to provide guidelines for the production of software for airborne systems that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. It is hoped that following the guidelines in DO-178B will ensure the production of reliable software that is documented, traceable, testable, and maintainable. The guidelines, however, do not stipulate specific reliability requirements for the software product since currently available reliability estimation techniques do not provide results in which confidence can be placed to the level required for certification purposes.

The DO-178B guidelines decompose the software life cycle into three major processes: a software planning process, software development processes, and integral processes. The software planning process defines and coordinates all of the project activities. The software development processes are those processes that actually produce the software product. These include the requirements, design, code, and integration processes. And finally, the integral processes ensure the correctness, control, and confidence of the software life cycle processes and their outputs. The integral processes consist of the verification, configuration management, quality assurance, and certification liaison processes.

To study the effectiveness of the DO-178B guidelines on the quality of the software, a simple case study in which two GCS implementations are being developed is being conducted. Two teams consisting of a programmer and a verification analyst have each been tasked to develop an implementation of the GCS following the DO-178B guidelines within the GCS test-bed. An extensive problem reporting system captures relevant software error information throughout the DO-178B development process. This data includes: a description of the software errors found; the activity when the error was detected, such as design review, unit testing, or integration testing; and, action taken with respect to the error. This data will allow us to not only look at the number of faults detected but, more importantly, the class of

faults found at different development stages and the relationship among the classes of faults found by the different verification techniques. This information coupled with the effort data for all development and verification activities could provide some insight into the effectiveness of the various development and verification methods.

After the two implementations have completed the DO-178B development process, the final software products will undergo testing in the simulated operational environment to help identify any remaining faults. These results could provide further insight into the effectiveness of the development methods and the reliability of the final software products.

Due to the extent of the data collection and configuration management procedures used in the test-bed, any phase in the life cycle of the GCS implementations can be reproduced. This gives a researcher the capability to go back to any one of the stages of the development process, apply a different development or verification technique to the software, and compare the resulting software to any previously developed implementation. Hence, the GCS development and verification environment can serve as a test-bed for the analysis of various software development and verification processes.

Many lessons have been learned about conducting software experiments during the course of this study of the DO-178B guidelines. A primary lesson is that a simple case study is not an adequate experiment design to evaluate an entire software development process. Conducting a more statistically rigorous software experiment, however, would require significant resources in terms of time and man-power. Development of the GCS test-bed, though, is a step toward conducting the experimentation necessary to provide the empirical data we need to scientifically evaluate and improve software processes and product quality.

The presentation provides further detail about the study of the DO-178B guidelines and the effort to conduct valid software experiments.

A Study of Software Standards Used in the Avionics Industry

Kelly J. Hayhurst
Assessment Technology Branch
Research and Technology Group

The Role of Computers in LaRC R&D Workshop
June 15, 1994

Outline

- **Background**
- **Software Standards**
- **Guidance and Control Software Project**
- **Summary**

Background

- Software is used in a wide variety of applications:
 - video games, answering machines, anti-lock brakes on cars, automatic teller machines, ..
- Software has many benefits compared to its hardware counterpart:
 - allows for more complex logic
 - provides increased flexibility
 - easier to modify
- Use of software is increasing in life- and safety-critical applications
 - avionics, Airbus 320
 - control of nuclear power plants

Software Engineering

- Software is a logical rather than a physical system element
 - Software is developed or “engineered” -- not manufactured

The establishment and use of sound engineering principles to economically obtain *reliable* software that works efficiently on real machines

- Engineering: the application of a systematic approach based on *science and mathematics*, toward the production of a product, process, or system

Reliable Software

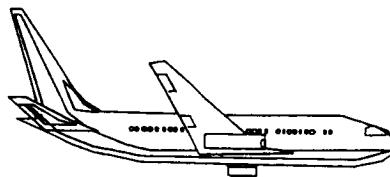
- Achieving reliable software is a global problem
 - no one knows how to generate perfect software
- Many proposed software reliability models (since '64)
 - Inadequate for estimation about life-critical software
 - most consider reliability growth based on faults found in development, as opposed to *operational reliability*
 - Often based on simplistic (unverified) assumptions
 - constant failure rates
 - stochastic independence
- Little existing data available to validate models

Software Dilemma

➔ Software can significantly expand system capability



➔ Since we don't know how to build perfect software -- Risk



How do we deal with these risks?

Software Standards

- There are a number of software guidelines/standards used in industry
 - DO-178B, used by the Federal Aviation Administration (FAA)
 - DoD-2167A, used by the Department of Defense
 - ISO 9000
- Provide the guidelines for the production of software that
 - performs its intended function
 - with some level of confidence that complies with the given requirements

Software Standards

- Many software development techniques, models and standards exist and are in use
 - most have been accepted largely based on logical arguments or anecdotal evidence

"...we need to codify standard practices for software engineering -- just as soon as we discover what they should be. Regulations uninformed by evidence, however, can make matters worse."

-- from Digital Woes (Why We Should not Depend on Software),
by Lauren Ruth Wiener

Focus

We need to become “...informed by evidence”

- **Conduct scientific experiments to understand:**
 - **software failure**
 - need to examine operational behavior of software
 - **the effect of different software development techniques**
 - relate that understanding to process models and standards

**Conduct Experiments!!
Collect Empirical Evidence!!**

Software Experiments in ATB

GOAL

Establish a controlled environment to conduct scientific experiments to address:

- ★ **the reliability of software and**
- ★ **the effectiveness of software development methods**

- **Guidance and Control Software (GCS) Project**
 - **study of the RTCA/DO-178B guidelines (Software Considerations in Airborne Systems and Equipment Certification)**
 - **“sponsored” by the FAA**

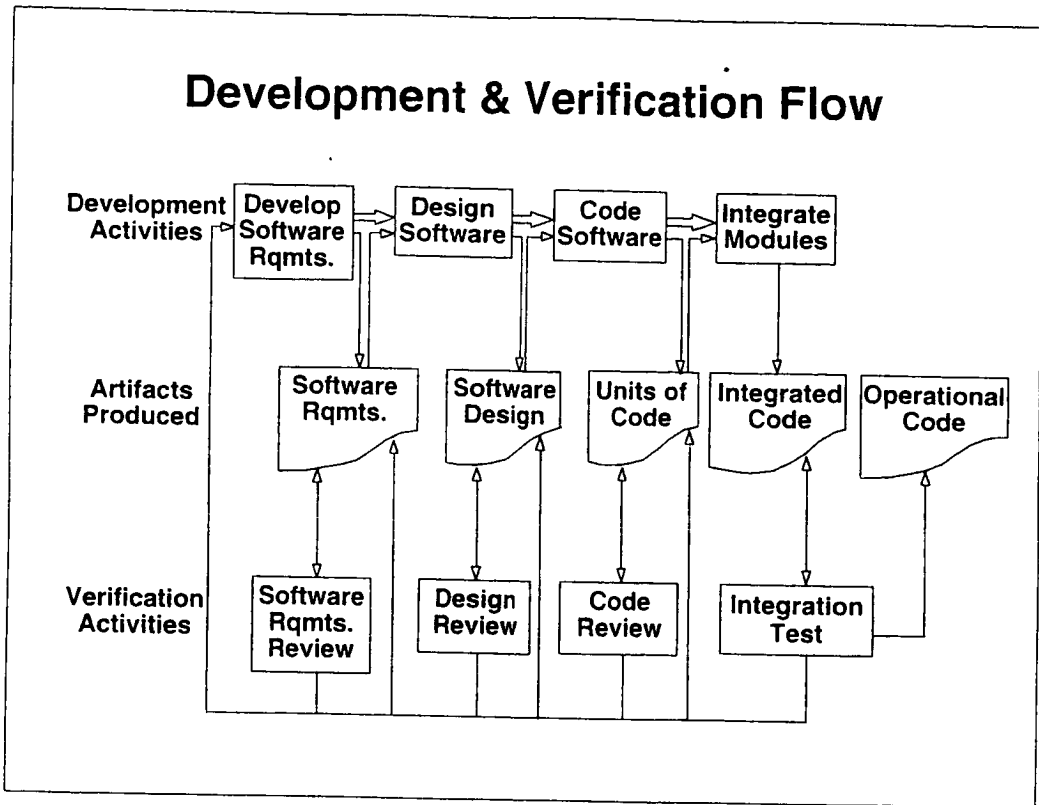
RTCA/DO-178B Guidelines

- **FAA requires compliance with DO-178B for software developed for embedded commercial aircraft equipment**
 - software designers must take a disciplined approach to software development
- **Gives general guidelines for software development and verification according to “software levels” -- A - E**
 - **A: anomalous behavior causes catastrophic failure condition**
 - **E: anomalous behavior has no effect on operational capacity**

Software Life Cycle Processes

- **Planning Process: defines and coordinates the software development activities**
- **Development Processes:**
 - Software Requirements Process
 - Software Design Process
 - Software Coding Process
 - Integration Process
- **Integral Processes: ensure correctness, control and confidence**
 - Software Verification Process
 - Software Configuration Management Process
 - Software Quality Assurance Process
 - Certification Liaison Process

Development & Verification Flow



DO-178B Life Cycle Data

Life Cycle Process	Life Cycle Data
Planning	Plan for Aspects of Certification Development Standards Accomplishment Summary
Development	Requirements Data Design Description Source Code Executable Object Code
Integral	Verification Plan Verification Procedures & Cases Verification Results Configuration Management Plan Configuration Management Records Development Environment Configuration Index Configuration Index Quality Assurance Plan Quality Assurance Records Problem Reports

CASE Tools

- **CASE tools can be used in the development of airborne software**
 - Any tool used must be qualified
- **Qualification is done by type:**
 - **Software Development Tools:** whose output is part of the airborne software
 - ex. source code generator
 - **Software Verification Tools:** tools that cannot introduce errors -- but may fail to detect them
 - ex. analysis of complexity tool

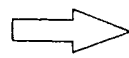
CASE Tools Qualification

- **For Software Development Tools:**
 - show that the development process used for the tool is equivalent to that used for the airborne software
- **For Software Verification Tools:**
 - show that the tool complies with its operational requirements under normal operating conditions

Study of DO-178B Guidelines

- Work with the FAA to evaluate methods that comply with the DO-178B guidelines
 - Base study on earlier work done at the Research Triangle Institute to study the DO-178A guidelines
- Experiment Design: One Shot Case Study

X O

 Apply DO-178B and see what you get

Guidance and Control Software Project

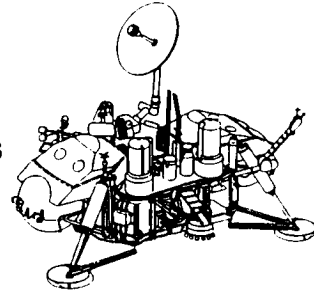
- Develop software according to DO-178B
 - use a guidance and control application
 - complete the life cycle starting from software requirements through integration
- Provide a controlled environment
 - extensive documentation and configuration control
 - extensive data collection
 - failure data
 - effort and cost data
- Simulate operation of the software to:
 - determine remaining faults
 - determine reliability

The GCS Application

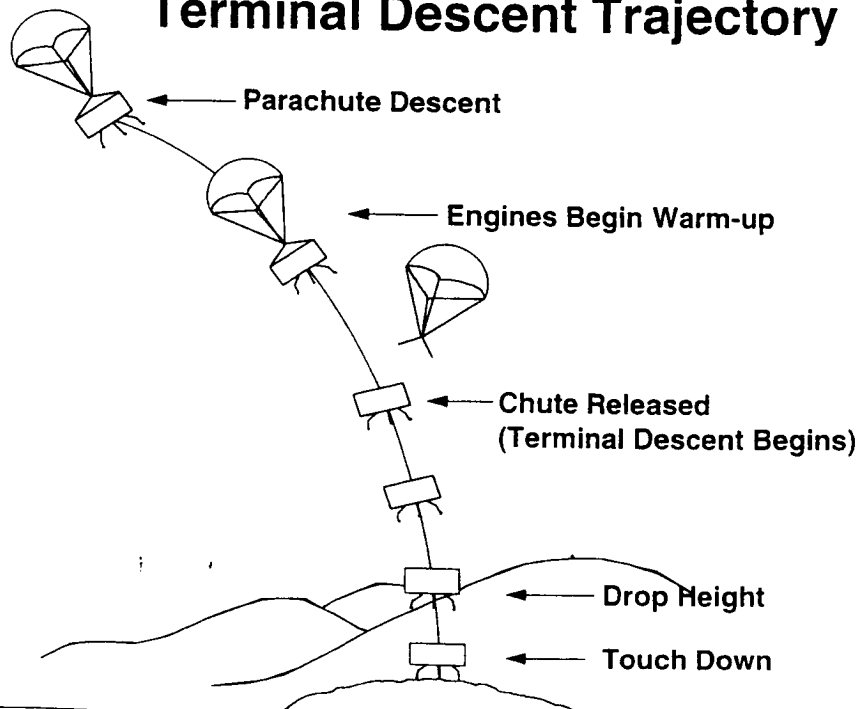
Purpose:

- (1) Provide guidance and engine control of a planetary landing vehicle during terminal descent to the planet's surface
- (2) Communicate sensory information about the vehicle and its descent to a receiving device

- Requirements are based on a simulation program used to study the probability of success of the 1976 Viking Lander mission to Mars



Terminal Descent Trajectory



Software Composition

The guidance and control software is composed of:

11 Functional Units which are divided into:

3 Subframes:

Sensor Processing
Guidance Processing
Control Law Processing

1 Frame = 1 iteration of the 3 subframes

1 Trajectory = ~ 2000 frames

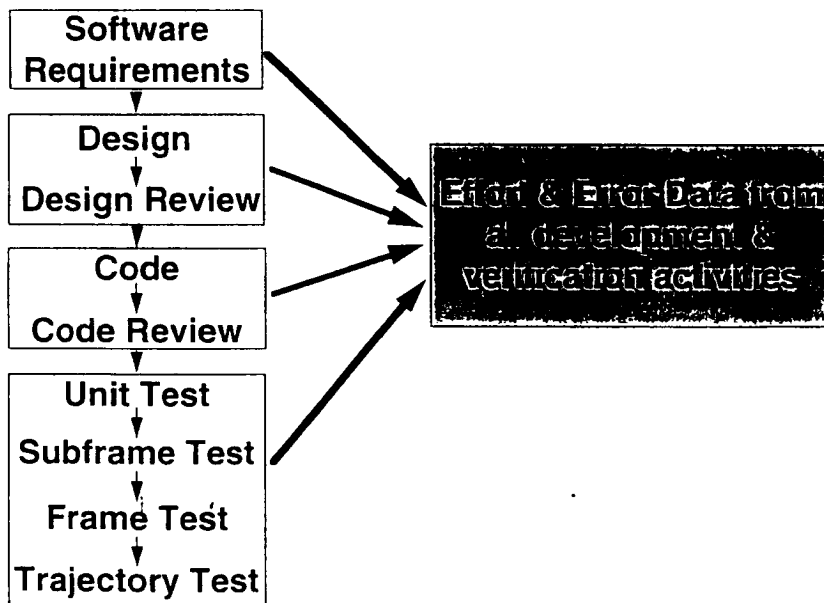
GCS Development Processes

- Producing 2 GCS implementations
 - each implementation has a designated programmer & verification analyst
- Each development team uses the same software high-level requirements document
- Designs generated using *teamwork*
 - conduct design review using formal inspection procedures
- Implementations coded in FORTRAN
 - projected size: 1500 - 2000 lines of code
 - conduct code review using formal inspection procedures

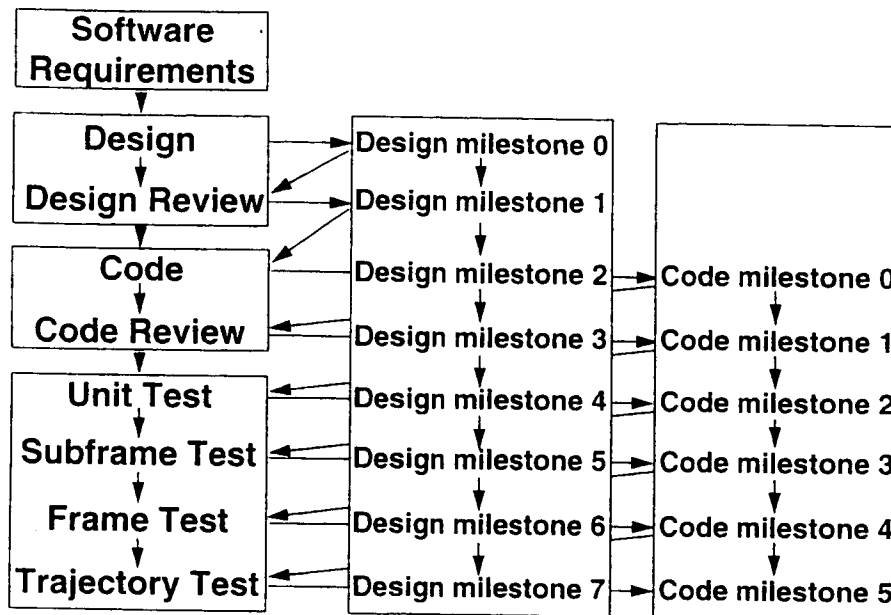
Integration Process

- Code is integrated at 4 levels: functional units
subframes
frames
trajectory
- Testing conducted at all 4 levels to:
 - demonstrate that the software satisfies its requirements
 - demonstrate (with high confidence) that errors which could lead to unacceptable failure conditions have been removed
- 100% coverage for requirements-based tests
- 100% modified condition/decision coverage

Development Products



More Products



Software Products

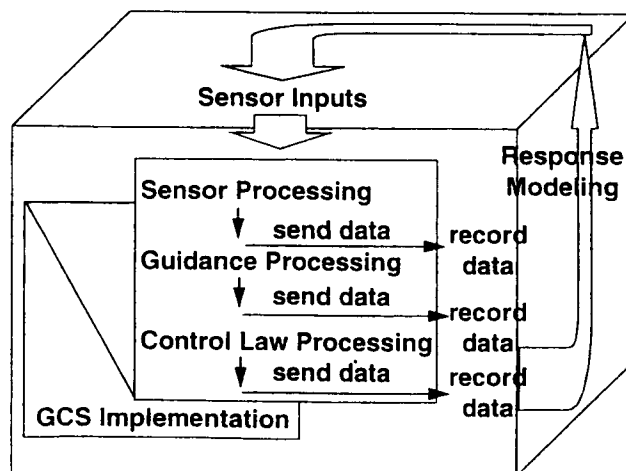
- Each software product (requirements, design, code, test cases, documentation) is placed under configuration control starting with the initial version
 - the Code Management System (CMS) by Digital Equipment Corp. is being used
- Each subsequent change to a software product is controlled and captured by the configuration management system
- All versions of any software product are preserved and can be reproduced

Experiment Basics

- Independently generate “n” implementations of the GCS
 - each following the development methodology defined in DO-178B
- Collect effort/cost data for all development and verification activities for each implementation
- Collect data on all faults identified in the software products throughout the development and verification processes
- Collect data on all faults identified in simulated operation

GCS Simulator

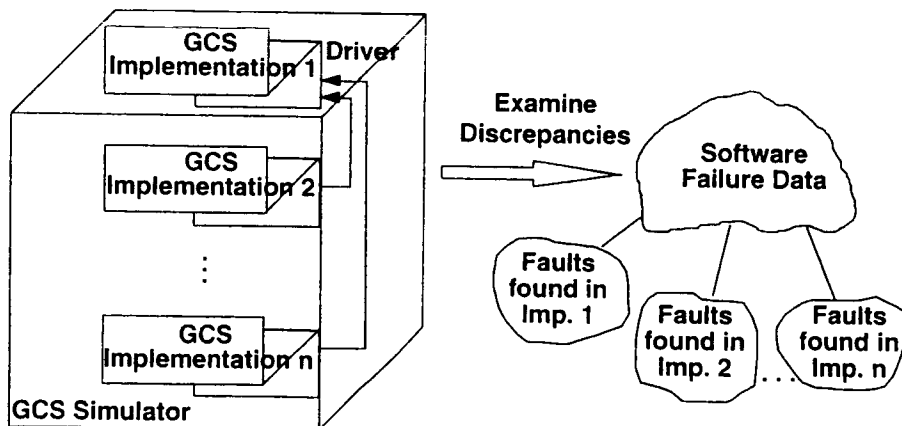
- Provides inputs (about environment & lander) for sensor processing
- Performs response modeling for the guidance and control
- Receives data



GCS Simulator

- Serves as a testbed for back-to-back testing of multiple GCS implementations (up to 28)
- For back-to-back testing, one implementation is designated as the “driver” implementation
- The results of all implementations are checked at the end of each subframe
 - for limit errors, comparing each variable against its predetermined valid range
 - for accuracy errors, comparing results of each implementation with results of the driver implementation
- All miscomparisons are recorded and investigated to determine the source of the problem

Operational Failure Characterization



- Use the software failure data to
 - estimate reliability of final version of each implementation
 - determine effectiveness of the development methodology

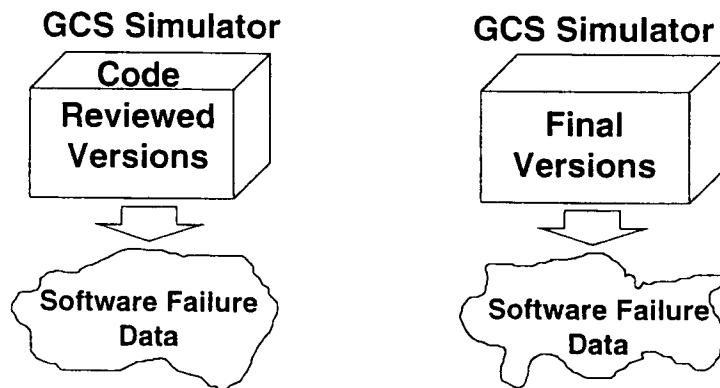
Understanding the Failure Data

Faults
found in
Implementation
n

Questions of Interest

- How many faults in the set?
- What types of faults?
- Are there any critical faults?
- Are there classes of faults found during random testing that are different than those found during DO-178B development cycle?

Studying Effectiveness



Are these fault sets equivalent?

- Is the integration process more effective (or efficient) compared to other fault detection methods?

GCS Project Status

- **The following project artifacts have been developed:**
 - Requirements for the guidance and control application
 - Configuration management system
 - GCS simulator
 - Data collection system
 - Project documentation
- **2 implementations are in the Design phase of development**
- **Plan to complete development by end of December '94**

Lessons Learned

- **Be prepared to document - and document -- and document**
- **Allow sufficient time up front for planning -- and documentation of those plans**
- **Tools can be helpful**
 - can help you organize and track items more efficiently
- **Tools can be hurtful**
 - it takes time (\$\$) to learn all about new tools and how to use them
 - allow for such time while planning
 - everyone involved with the output of a development tool needs to understand that tool

More Lessons

- **Complying with the DO-178B guidelines is not cheap**
 - developing critical software is time, man-power, and documentation intensive
- **Collecting data -- software failure data and cost/effort data -- is difficult**
 - software problems are often complex
 - changes can impact many project artifacts
 - reluctance to accurately account for development effort

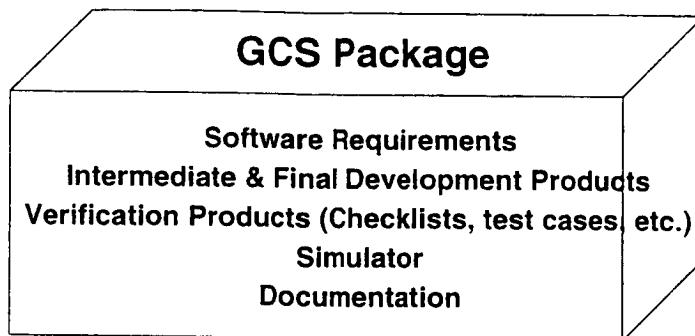
Summary

**Gathering empirical evidence is difficult
-- But IMPORTANT!!!**

- **GCS project provides a controlled environment to observe and collect empirical data on software development methods**
 - Realistic guidance and control application
 - Applying industry-standard guidelines and practices
- **Provide data to increase understanding of software development processes and the quality of their products**
 - improve software processes & product quality
 - improve reliability estimation methods
 - provide input for improving software standards

Project Plans

- Make the GCS testbed available to other researchers
- Improve the experiment design to allow more statistical analysis



A Software Tool for Dataflow Graph Scheduling

Robert L. Jones III
NASA Langley Research Center

P. 24

A graph-theoretic design process and software tool is presented for selecting a multiprocessing scheduling solution for a class of computational problems. The problems of interest are those that can be described using a dataflow graph and are intended to be executed repetitively on multiple processors. The dataflow paradigm is very useful in exposing the parallelism inherent in algorithms. It provides a graphical and mathematical model which describes a partial ordering of algorithm tasks based on data precedences. That is, some tasks must execute in a particular order whereas other tasks may execute independent of other tasks. Dataflow graph nodes represent schedulable tasks and edges represent the data dependencies between the tasks. Analytical analysis of the dataflow graph is possible for many digital signal processing (DSP) and control law algorithms which are deterministic. For determinism, the model is applicable to a class of dataflow graphs where the time to execute tasks are assumed constant from iteration to iteration when executed on a set of identical processors. Also, it is assumed that the dataflow graph is data independent. Any decisions present within the computational problem must be contained within the graph nodes rather than described at the graph level. Special transitions called sources and sinks are also provided to model the input and output data streams of the task system. The presence of data is indicated by marking the dataflow graph with tokens. The graph transitions through markings as a result of a sequence of node firings. A node is enabled for firing when a token is available on every input edge of the node, indicating that the task has all of its operands. When the node fires, it encumbers one token from each of its input edges, delays an amount of time equal to the task latency, and then deposits one token on each of its output edges. Sources and sinks have special firing rules in that sources are unconditionally enabled for firing and sinks consume tokens, but do not produce any. By analyzing the dataflow graph in terms of its critical path, critical circuit, dataflow schedule, and the token bounds within the graph, the performance characteristics and resource requirements can be determined *a priori*.

As for any mathematical model, there is a need for efficient software tools which facilitate the use of the model in solving problems. A software program, referred to as the Dataflow Design Tool, was developed at Langley to apply the dataflow model and design multiprocessor solutions for spaceborne applications. The tool was written in C++ for Microsoft Windows 3.1 or NT can be hosted on an i386/486 personal computer or compatible. The Design Tool takes input from a text file which specifies the topology and attributes of the dataflow graph. A Graph Tool was developed to facilitate the creation of the graph text file. The various displays and features are shown to provide an automated and user-interactive design process which facilitates the selection of a multiprocessor solution based on dataflow analysis. Performance metrics determined automatically by the Dataflow Design Tool include the minimum time to execute all tasks for a given computation (schedule length), the minimum time between graph input and the corresponding output (TBIOb), the minimum graph-imposed iteration period (T_0), and the minimum time between outputs (TBOlb). Also, the tool determines if tasks can be delayed a finite amount of time without degrading performance, referred to as slack time. Since the edges imply the physical storage of data, the tool can calculate the minimum data buffers required for proper

sharing of data between tasks. In addition to numerical performance metrics, the tool graphically portrays system behavior using Gantt charts and resource envelopes. The Single Graph Play displays the steady-state task schedule associated with a single computation, and the Total Graph Play displays the periodic, steady-state task schedule over a single iteration period.

The analysis and multiprocessor mapping of a finite impulse response (FIR) filter is illustrated. A linear phase FIR filter is selected since it requires half the number of multiplies of other FIR realizations. DSP problems are very suitable for dataflow analysis since they are typically described as signal flow graph. One can easily translate signal flow graphs to dataflow graphs by locating the computations (addition and multiplication) and representing unit delays (inverse z terms) with initial tokens. Once the filter has been captured into the Graph Tool it can be analyzed by the Dataflow Design Tool to expose the inherent parallelism and determine graph-theoretic performance bounds. Since there are many realizations of the same filter, characterized by different dataflow graphs, the Dataflow Design Tool can be useful in determining which realization exposes the most parallelism. The SGP shows that some of the additions can execute in parallel (C1 through C4), enabling the parallel execution of the multiplies, and finally the sequential summation to generate the output sample. The SGP bars are shaded to depict the read, process, and write activities of the processor, and the hollow bars denote slack time associated with some tasks. In addition to the parallel concurrency, the TGP shows pipeline concurrency that may be exploited. In this example, the TGP shows that at most 4 different data samples may be computed within a sampling period of 224 time units. The Total Resource Envelope shows that 10 processors are required to achieve this level of throughput. The dataflow analysis applied to the dataflow graph and portrayed in the graph play diagrams assume infinite resources (processors and memory) so that the exposed parallelism is limited only by the data precedences. If there is not enough resources to exploit the inherent parallelism, the schedule must be optimized. As an example, let's assume that a fully-static schedule (i.e., a task will execute on the same processor for every iteration) on 8 processors is desirable to minimize run-time overhead. The Dataflow Design Tool shows that such a solution can be achieved by inserting two additional "artificial" data dependencies and increasing the sampling period to 260 time units. The tool can also display the periodic memory accesses within a periodic schedule. Such an assessment may be useful to optimize the schedule based on the limited bandwidth between processors or processors and memory. Once a desirable solution is obtained, the tool can summarize the scheduling constraints in terms of earliest start (ES), latest finish (LF), and slack time. The summary of run-time requirements include task instantiations (INST) defined as the number of processors a task will have to execute on simultaneously for different data sets. For a fully-static schedule, one would expect all instantiations to be 1 as shown. Also, the buffer sizes (QUEUE) for shared data is given along with the number of initially empty buffers (OE) and the number of initially full buffers (OF) due to initial data.

In summary, the dataflow paradigm provides a general model suitable in exposing parallelism inherent in algorithms as fine-grain as filters to more computationally complex algorithms where a node might represent an entire filter. When the algorithm is deterministic, the Dataflow Design Tool can analytically determine the steady-state behavior, performance bounds, scheduling constraints, and resource requirements. By permitting the user to insert artificial data dependencies, the dataflow schedule can be optimized to match resource requirements with resource availability.

A Software Tool for Dataflow Graph Scheduling



June 15, 1994

Robert L. Jones III

*NASA Langley Research Center
Hampton, Virginia*

Outline

- Functional Overview
- Analysis of a DSP Filter
- Static Scheduling and Optimization
- Summary

Dataflow Design Tool



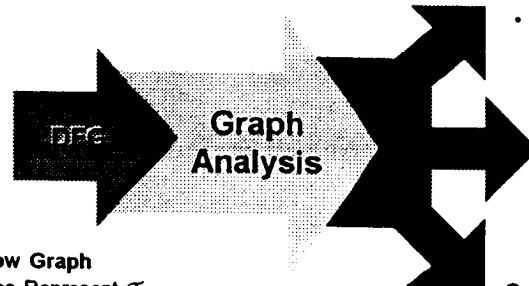
Task System, $(\mathcal{T}, \mathcal{L}, \prec, \mathcal{M}_0)$

- \mathcal{T} Set of Tasks
- \mathcal{L} Fixed Task Latencies
- \prec Partial-Order on \mathcal{T}
- \mathcal{M}_0 Initial State

⇒ Dataflow Graph (DFG)

Performance Bounds

- Schedule Length, ω
- Time Between Input & Output, T_{BIO_b}
- Minimum Iteration Period, T_0
- Time Between Outputs, T_{BO_b}
- Slack
- Processor Utilization



Run-Time Requirements

- Task Instantiations
- Processor Requirement
- Data Buffers
- Artificial \prec , Control Edges

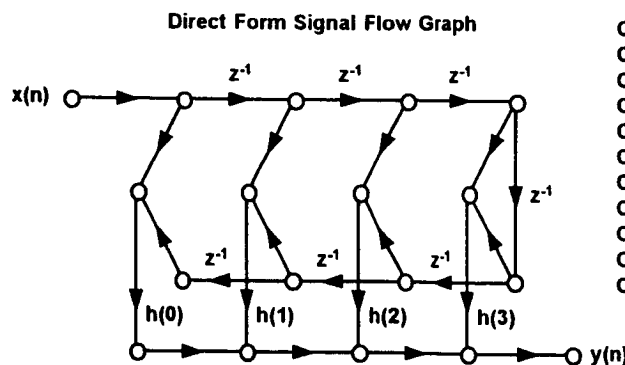
Dataflow Graph

- Nodes Represent \mathcal{T}
- Edges Describe \prec
- Tokens Indicate Presence of Data
- Initial Marking = \mathcal{M}_0

Graphical Displays

- Gantt-Chart Task Execution
 - Single Iteration (SGP)
 - Periodic Execution (TGP)
- Resource Envelopes

Eight-Order, Linear Phase FIR Filter



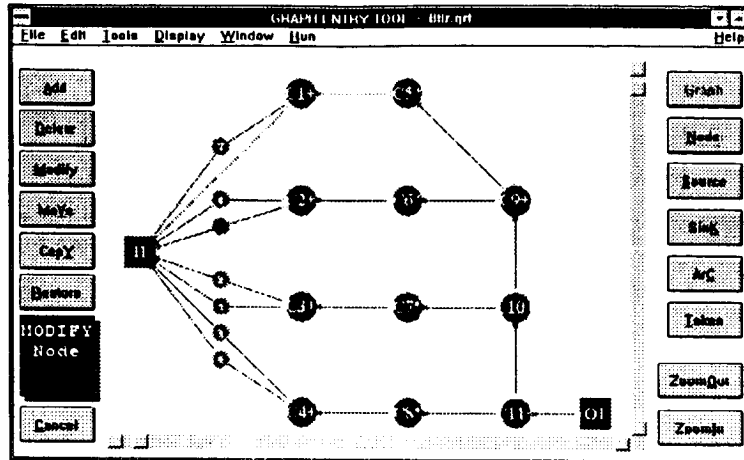
Tasks Instructions

- C1+ $x_0 = x(n) + x(n-7)$
- C2+ $x_1 = x(n-1) + x(n-6)$
- C3+ $x_2 = x(n-2) + x(n-5)$
- C4+ $x_3 = x(n-3) + x(n-4)$
- C5* $x_4 = x_0 * h(0)$
- C6* $x_5 = x_1 * h(1)$
- C7* $x_6 = x_2 * h(2)$
- C8* $x_7 = x_3 * h(3)$
- C9+ $x_8 = x_4 + x_5$
- C10+ $x_9 = x_6 + x_8$
- C11+ $y(n) = x_7 + x_9$

A DSP signal flow graph is a *Dataflow Graph* where the z^{-1} unit delays can be modeled with initial tokens. Thus, run-time implementation of *delay* does not incur any overhead. Unit delays are simply implemented by initializing FIFO queues used for intermediate data.

Dataflow Graph Capture of FIR Filter

Graph Tool



Multiprocessor Implementation Example

Assumptions: Shared memory with no contention
 Multiplies take 200 time units
 Additions take 100 time units
 One-operand read/writes take 10 time units
 Two-operand read/writes take 20 time units

Performance

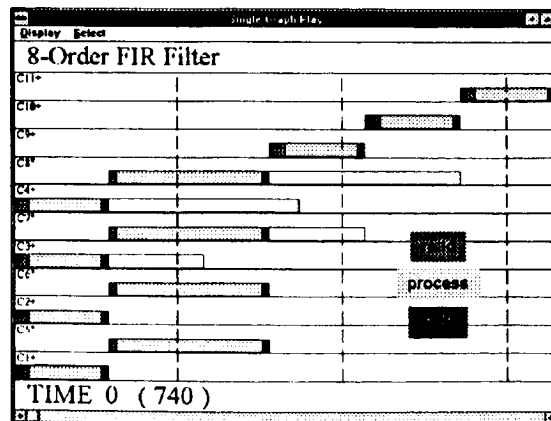
Display	Get
Graph	
TCE	1730
TMOB	740
TMOA	274
Multiplies	740
TBO	274
Processors	8

Division of Effort

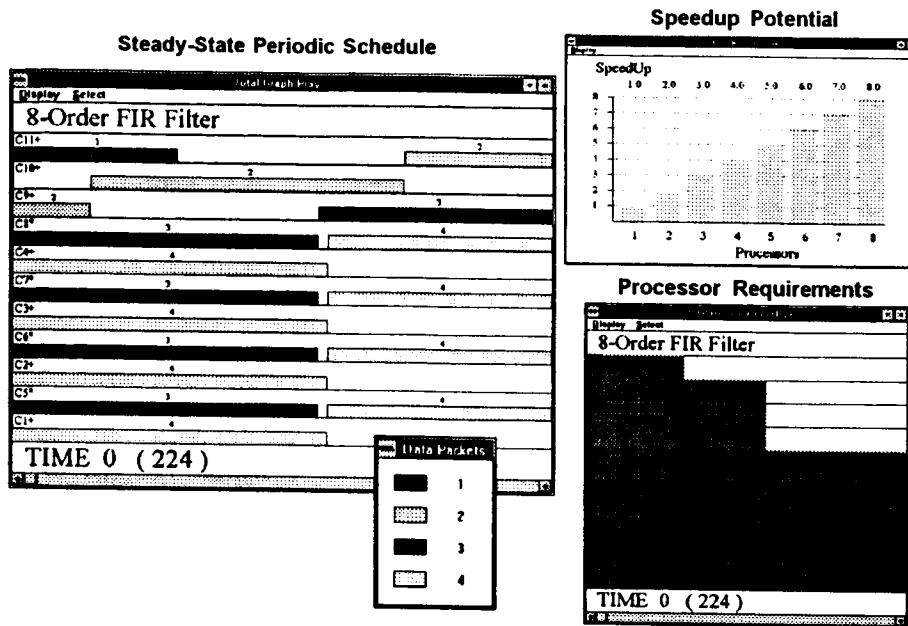
Total Computing Effort	
Processing	= 1500
Read/Write	= 290
Overhead	= 15.2 %

OK

Data-Driven Schedule for One Iteration

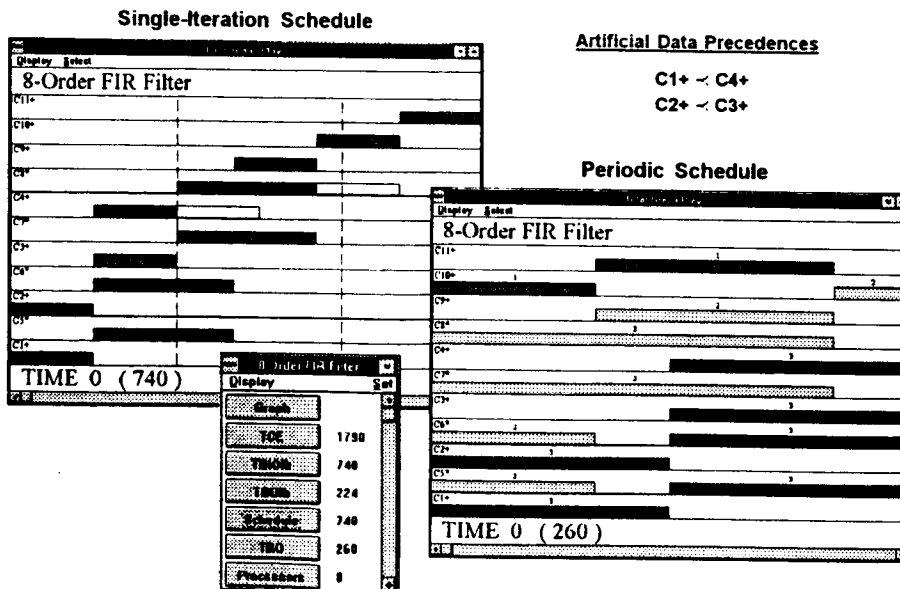


Exposing the Parallelism in the FIR Filter



Optimization for 8 Processors

A fully-static schedule is desired for minimum run-time overhead.



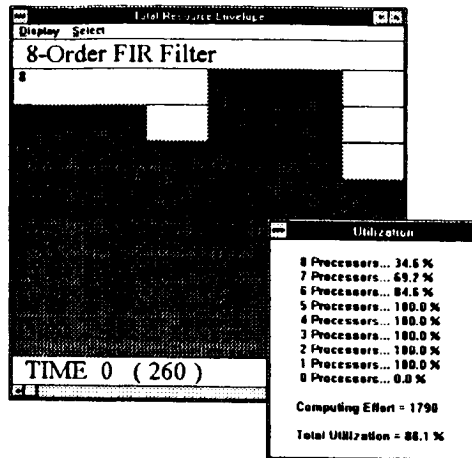
Fully-Static Processor Requirements

Sampling Period = 260 time units

Processor Assignments

- P1 (C1+, C4+)
- P2 (C2+, C3+)
- P3 (C5*)
- P4 (C6*)
- P5 (C7*)
- P6 (C8*)
- P7 (C9+, C10+)
- P8 (C11+)

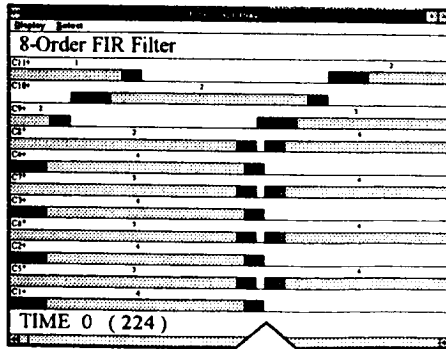
Total of 8 DSP Chips are Required



Analysis of Memory Access

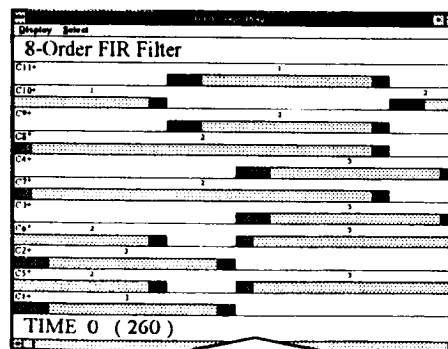
Optimized schedule has better distribution of memory accesses which e.g., can be accommodated with 6 independent communication ports in the TMS320C40's.

Unoptimized Schedule



Too many localized memory references!

Optimized Schedule



Memory references are more evenly distributed.

Summary of Fully-Static Multiprocessor Solution FIR Filter

NAME	LATENCY	ES	LF	SLACK	INST	DEJOF	QUEUE
Π						1/4 → C4+	5 → C4+
						1/5 → C3+	6 → C3+
						1/6 → C2+	7 → C2+
						1/7 → C1+	8 → C1+
						1/3 → C4+	4 → C4+
						1/2 → C3+	3 → C3+
						1/1 → C2+	2 → C2+
						1/0 → C1+	1 → C1+
C1+	130	0	130	0	1	1/0 → C6+	1 → C4+
						1/0 → C5+	1 → C5+
C5+	220	130	350	0	1	1/0 → C8+	1 → C8+
C2+	130	0	130	0	1	1/0 → C3+	1 → C3+
						1/0 → C6+	1 → C6+
C6+	220	130	350	0	1	1/0 → C9+	1 → C9+
C3+	130	130	260	0	1	1/0 → C7+	1 → C7+
C7+	220	260	480	0	1	1/0 → C10+	1 → C10+
C4+	130	130	260	130	1	1/0 → C8+	1 → C8+
C8+	220	260	480	130	1	2/0 → C11+	2 → C11+
C9+	130	350	480	0	1	1/0 → C10+	1 → C10+
C10+	130	480	610	0	1	1/0 → C11+	1 → C11+
C11+	130	610	740	0	1	1/0 → O1	1 → O1

Summary

- Dataflow provides a general model of computation capable of exposing fine- and large-grain parallelism.
- Design Tool provides analytic, compile-time prediction of:
 - Steady-state behavior
 - Graph-theoretic performance bounds
 - Iterative run-time scheduling criteria
- Permits inclusion of artificial precedences for optimization.
- Facilitates selection of static run-time schedules.

Use of Software through Pictures on CERES

The CERES team has been using the Yourdon/DeMarco Structured Analysis/Structured Design methodology to develop the data management system for producing higher order science data products from CERES instrument data. As part of this effort, the team is using the Software through Pictures CASE tool to automate portions of the methodology. This presentation addresses the team's experiences with the selected methodology and CASE tool, describes lessons learned, and provides recommendations for other teams contemplating the use of structured methodologies and CASE tools.

Software Engineering methodologies can help developers create systems in less time with higher reliability and quality by providing tools for managing the complexity inherent in software systems and development programs. CASE tools can facilitate using a methodology by providing tools for creating and maintaining requirements and design models, automating consistency and completeness checking, and automating much of the bookkeeping associated with following the methodology. This allows developers to focus on the creative aspects of software design and development.

Overall, our experience on CERES has been that structured methodologies and CASE tools prove useful in creating, maintaining, and documenting high quality requirements and analysis products. Although the learning curves associated with these tools require an investment in time and training early on, the benefits to be gained are well worth the effort and our productivity continues to increase as we become more familiar with the methodology.

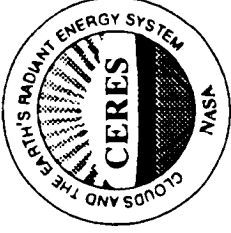
To date, the CERES data management team has used the tool to model more than 130 data products down to the level of atomic variables, define each data element in terms of type, units, accuracy, and number of bits, and create documentation from the information stored in the models. Since the CERES system is primarily a science data processing system which generates more than 5 terabytes of data per month, focusing on the system's data products has led to a deeper understanding of processing needs and resulted in higher quality functional requirements. Furthermore, the graphical editors and consistency checking features provided by the tool have allowed the team to rapidly iterate through the modelling process in less time than would have been required without the tool.

The data management team is currently analyzing system functional requirements by modelling the functionality needed to process instrument and higher order science data. Here again, the tool speeds up the process of iterating on the model to converge on a final solution. In addition, the tool has allowed the team to automatically produce software requirements documents in a standard format from information contained in the CASE tool database.

We have incorporated several customizations in order to tailor the CASE tool to support the specific processes employed on CERES. These customizations include creating templates for producing CERES-specific documentation, enhancing the CASE tool main menu, and integrating the CASE tool with the FrameMaker desktop publishing package. The CASE tool is supplied with templates for producing documentation that complies with military software standards. Since these standards were not appropriate for NASA publications, we developed templates for

several documents including a Software Requirements Document, Data Products Catalog, and Data Dictionary as well as several utilities to provide hard copies of details stored in the tool's database for developer's use in reviewing their models. We have also modified the tool's main menu to simplify the user interface for creating documents. Finally, there are several places in the tool where the developer adds detail to the requirements or design model by entering free form text. These items include functional descriptions, data product descriptions, and interface descriptions. The CASE tool only supports ASCII text and, since much of our processing is described in terms of equations, tables, and graphics this restriction limited our ability to fully describe the necessary processing. Therefore, we have modified the tool to allow the use of FrameMaker (desktop publishing/word processor) for entering descriptions of functions, data products, and interfaces. This allows a designer to include any combination of text, graphics, tables, and equations in these descriptions which are then included directly into the documentation produced using the tool.

Our experience indicates that when combined with well-structured methodologies, CASE tools can provide an important component of a development environment which helps designers create software products with higher quality in less time. However, the key to achieving productivity gains is the process used to design the software. The processes incorporated in structured analysis and structured design provide a sound framework for creating complex software systems and must be adopted in order to derive any benefits from the use of automated tools such as Software through Pictures.



Use of Software through Pictures on CERES

**The Role of Computers in LaRC R&D Workshop
June 15-16, 1994**

**Troy Anselmo
Science Applications International Corporation**



INTRODUCTION

- **CERES Overview**
- **Software Development Methodology**
- **CASE Tool Capabilities and Configuration**
- **Experiences to Date**
- **Lessons Learned/Recommendations**

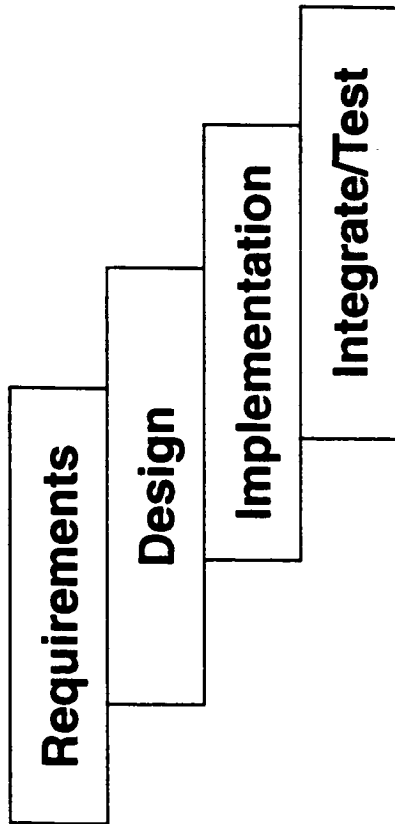


CERES OVERVIEW

- **Scientific Data Processing Application**
- **Approximately 500K Source Lines of Code**
- **Organized into 12 Subsystems (CSCIs)**
- **Generates More Than 5 TeraBytes of Data per Month**
- **Operates within the EOIS Environment**
- **Languages include FORTRAN, C, Ada**

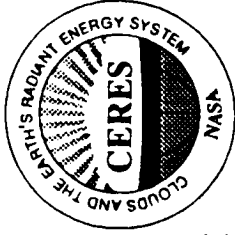


METHODOLOGY



Build the Right Product

Build the Product Right

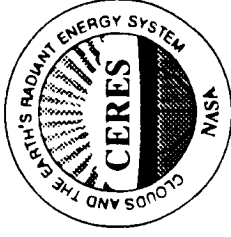


- Methodology Consists of:
- Notation - Capture Models, Reason about Models, Communicate to Others
- Process - Procedures for Creating Models

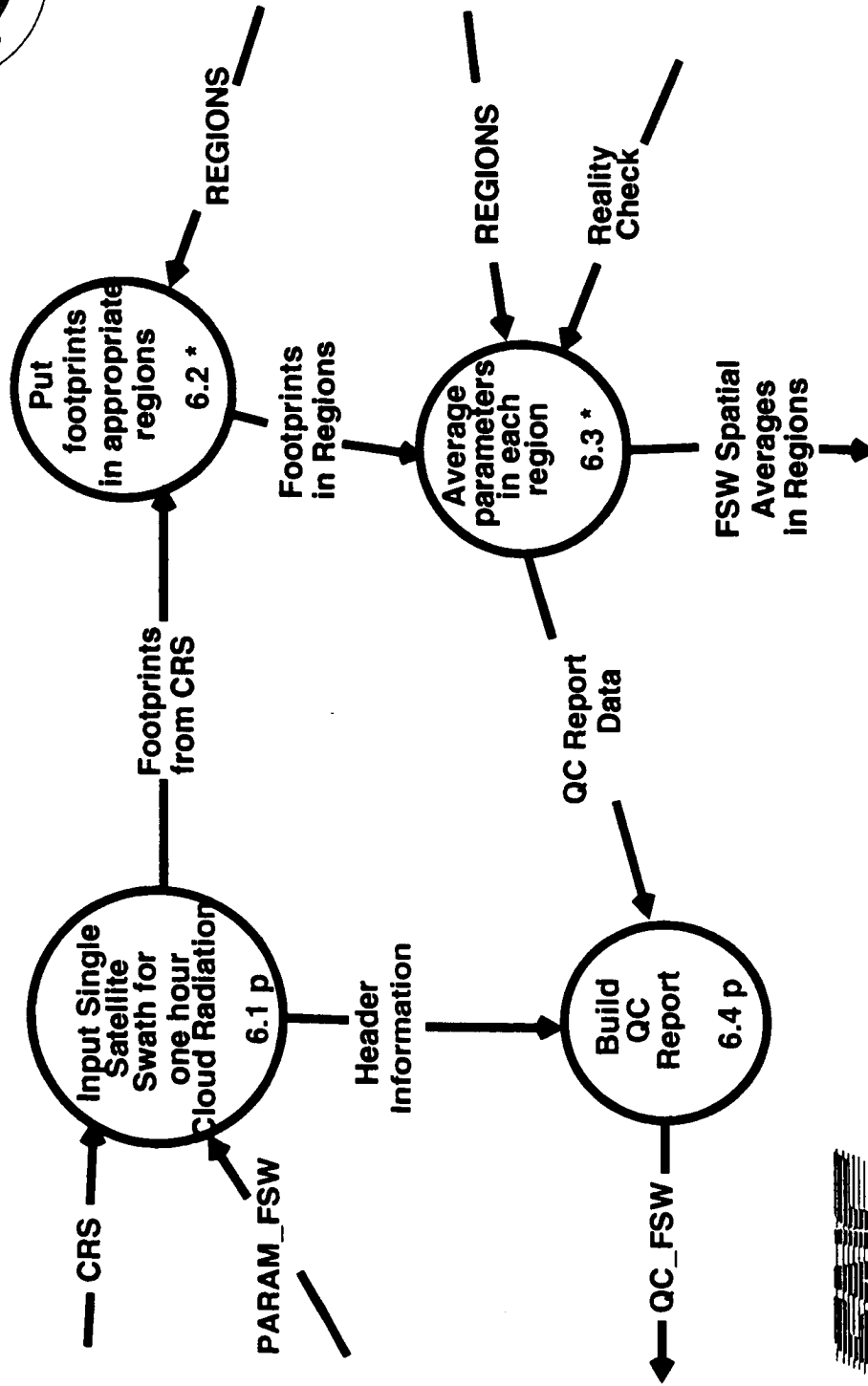
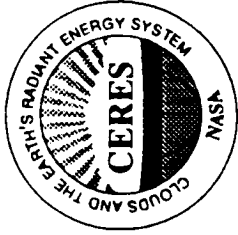


METHODOLOGY (cont'd)

- **Structured Analysis/Structured Design**
- **Model Based Approach**
- **Emphasis on Early Life Cycle Phases**
- **Requirements - Model functionality**
- **Design - Models Architecture of Solution**

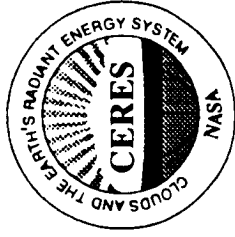


METHODOLOGY (cont'd)

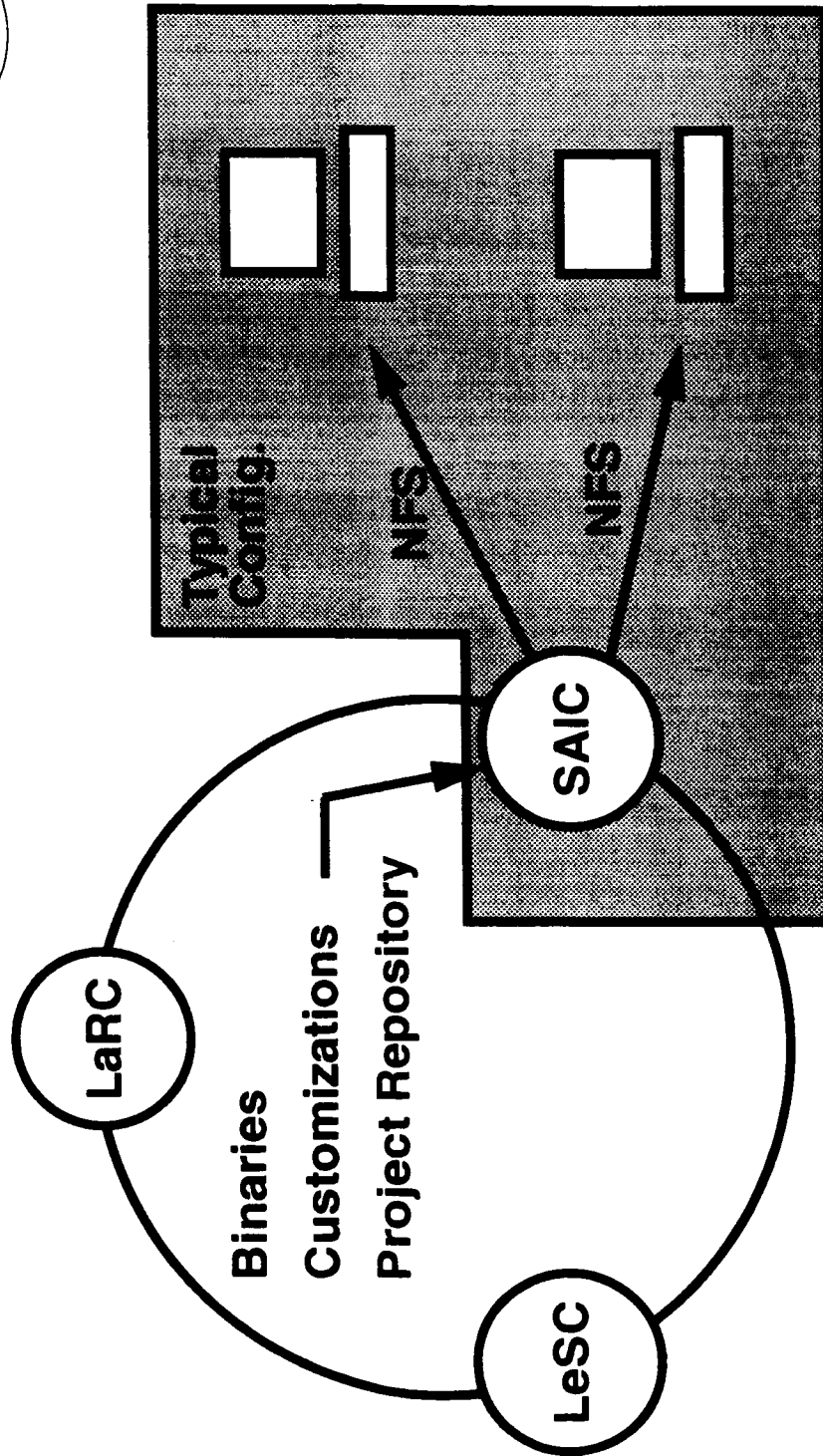


CASE TOOL CAPABILITIES

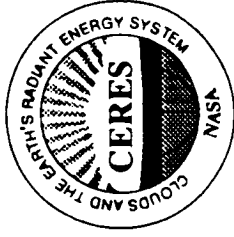
- Automate Portions of Methodology so Developers can Focus on Creative Aspects of Software Design
- Provide Tools to:
 - Rapidly Create and Modify Models
 - Capture Models in Central Repository
 - Check Model Validity (Completeness, Consistency)
 - Support Multiple Developers in Work Group Environment
 - Create Documentation from Models in Repository



CASE TOOL CONFIGURATION



EXPERIENCES TO DATE



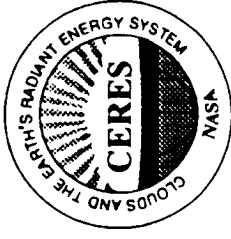
- **Achievements**
- **Data Products Modelled (incl. Data Structure and Data Description Details)**
- **Data Product Catalogs Generated from Data Models (Sizing Analysis Computed by Tool)**
- **Currently Modelling Each Subsystem**
- **Automatically Produce Requirements Documentation in Standard Format for Each Subsystem**

EXPERIENCES (cont'd)

- Customizations
- CERES Specific Document Templates
- Main Menu Changes to Facilitate Document Generation
- Integration with FrameMaker to support Graphics, Tables, and Equations
- Positive Results
- + Validation of Interfaces Between Subsystems
- + Communication of Functional and Data Models Among Team Members (incl. Science Team)
- + Allows Rapid Iterations on Diagrams to Converge on Solutions



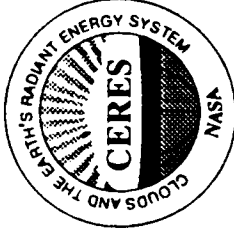
EXPERIENCES (cont'd)



- **Issues**
 - **Multiple-Site Configuration Complicated System Administration Functions**
 - **Document Definition in Parallel with Template Development Resulted in Excessive Template Iterations**
 - **“Loose” Configuration Management of Customizations Created Synchronization Problems Among Multiple Sites**
 - **Lack of CASE/Methodology Expertise at Each Site Slowed CASE Tool Adoption**



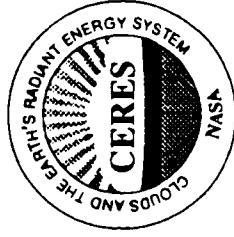
EXPERIENCES (cont'd)



- **Quality Action Team Established by NASA to Address Concerns, Improve CASE Tool Use**
- **Membership From All Organizations**
- **Results to Date Have Been Very Positive**
- **Enhanced Understanding and Awareness of Concerns Among Organizations**
- **Simplified System Administration Process and Configuration**
- **Establish Forums for Information Dissemination, Identified Training Needs, Conducted Training**
- **Improved Development, Test, CM Process for Customizations**



LESSONS LEARNED/RECOMMENDATIONS



- **CASE Tool Introduction Represents Potential Culture Change**
- **Strong Management Support Required**
- **Steering Committee Useful for Coordinating Adoption Process**
- **Timely Dissemination of Information Necessary, Exploit Electronic Communications Media (e-mail, bulletin boards, WWW)**
- **Methodology is Key Element, Training is Critical**
- **CASE is Engineering Tool, Documentation is By-Product**
- **Customizations Represent Development of Utility Codes, Should Use Structured Development Process**



SESSION 4 Solutions of Equations

Chaired by

Olaf Storaasli

- 4.1 Rapid Solution of Large-scale Systems Of Equations - Olaf Storaasli
- 4.2 Solution of Matrix Equations Using Sparse Techniques -Majdi Baddourah
- 4.3 Equation Solvers for Distributed Memory Computers - Olaf Storaasli

Rapid Solution of Large-scale Systems Of Equations

by Olaf O. Storaasli, (O.O.Storaasli@larc.nasa.gov or 804-864-2927)

for Workshop on the Role of Computers in Langley R&D (6-15-94)

The analysis and design of complex aerospace structures requires the rapid solution of large systems of linear and nonlinear equations, eigenvalue extraction for buckling, vibration and flutter modes, structural optimization and design sensitivity calculation. Computers with multiple processors and vector capabilities can offer substantial computational advantages over traditional scalar computers for these analyses. These computers fall into two categories: shared-memory computers (e.g., Cray C-90) and distributed-memory computers (e.g., Intel Paragon, IBM SP-2).

Shared-memory computers have only a few processors (16 on a Cray C-90), which rapidly process vector instructions (simultaneous adds and multiplies) and address a large memory. Information is shared among processors by referencing a common variable in shared-memory.

Distributed-memory computers may have thousands of processors, each with limited memory. Explicit message passing commands (i.e. send, receive), are used to communicate information between processors. Such communication is time consuming, so algorithms need to be designed to run efficiently on distributed-memory computers.

This presentation will cover general-purpose, highly-efficient algorithms for: generation/assembly of element matrices, solution of systems of linear and nonlinear equations, eigenvalue and design sensitivity analysis and optimization. All algorithms are coded in FORTRAN for shared-memory computers, and many adapted to distributed-memory computers. The capability and numerical performance of these algorithms will be addressed.

O. Storaasli, D. Nguyen, M. Baddourah and J. Qin (1993), "Computational Mechanics Analysis Tools for Parallel-Vector Supercomputers", AIAA/ASME/ASCE/AHS/ASC 34th Structures, Structural Dynamics and Materials Conference Proceedings, Part 2, pp. 772-778 (Int. J. of Computing Systems in Engineering, Vol 4, No. 2-4, 1993)

Dr. Olaf Oliver Storaasli is a senior research scientist in computational mechanics at the NASA Langley Research Center, Hampton, Virginia. He began his career at Langley after receiving a Ph.D. degree in Engineering Mechanics from North Carolina State University in 1970.

Long before parallel computers were commercially available, Dr. Storaasli led a hardware, software and applications team at NASA Langley Research Center to develop one of the first parallel computers, the Finite Element Machine. He has authored over 80 works in computational structural mechanics including static and dynamic structural analysis, eigenvalue and optimization methods, interdisciplinary analysis, data management, and parallel-vector structural analysis methods on supercomputers. He received the Floyd L. Thompson Fellowship of NASA Langley Research Center for post-doctoral research at Norges Tekniske Hogskole in Trondheim, Norway, and Det Norske Veritas, Oslo, Norway, during 1984-85 and has been invited back twice since. He received 5 NASA-wide and 8 Langley Achievement awards for outstanding work in Computational Structural Mechanics. These awards included significant contributions to the NASA Viking and Integrated Programs for Aerospace-Vehicle Design (IPAD) Projects as well as to the development of Relational Information Management (RIM), since developed into the commercial relational data-base software: R:BASE. In August, 1989, Cray Research selected the general-purpose matrix equation solution software, pvsolve, developed by Dr. Storaasli and his colleagues, to receive the GigaFLOP Performance Award. pvsolve was used to solve the 54,870 equations (9.2 billion floating point operations) in the Space Shuttle Solid Rocket Booster structural analysis in six seconds elapsed time. His recent research has resulted in methods to analyze a 172,400 equation (5,737 bandwidth) refined model of a high speed civil transport and a 265,000 equation automobile (3,374 bandwidth) application in less than two minutes on the Cray C-90 and a method to generate and assemble structural stiffness matrices on the Intel Delta at speeds 25 times that of one Cray C-90 processor.



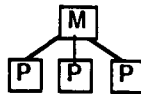
Rapid Solution of Large Systems of Equations



Dr. Olaf Storaasli

Computational Structures Branch
Mail Stop 240
NASA Langley Research Center
Hampton, VA 23681

Email: O.O.Storaasli@larc.nasa.gov
Phone: 804-864-2927
FAX: 804-864-8912

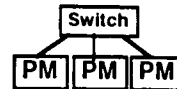


presented at

Workshop on

The Role of Computers in Langley R&D

June 15, 1994, Reid Conference Center
NASA Langley Research Center



Langley
Research
Center
00-1

Objective

- ***Faster, cheaper, better* analysis/design of large-scale structures**
 - Develop algorithms to exploit high-performance computers
 - Evaluate computational performance



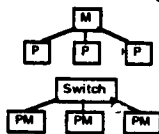
Langley
Research
Center
00-2



Outline



- Supercomputers & Structural Models
- Structural Analysis
 - Nodal Generation and Assembly
 - Linear Equation Solvers



Shared-memory computers

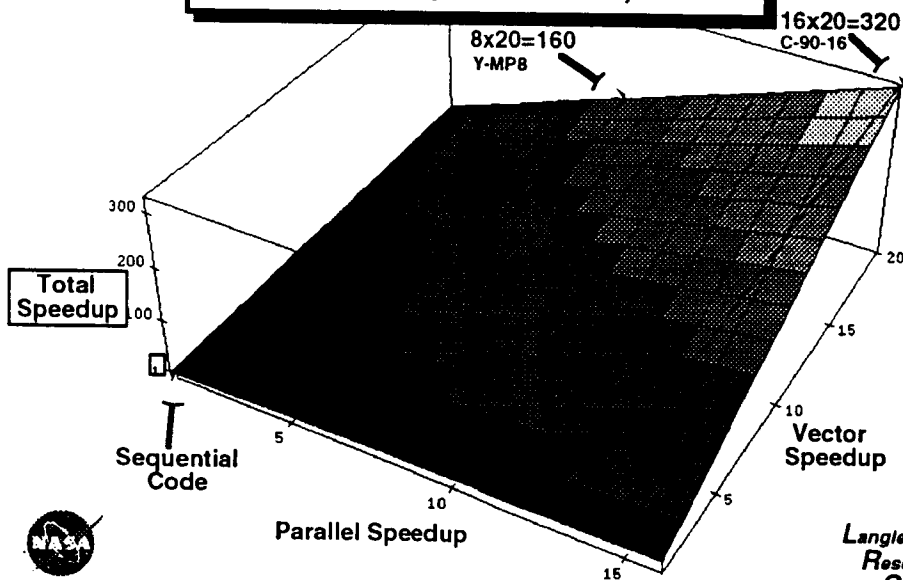
Distributed-memory computers

- Nonlinear Equation Solvers
- Structural Optimization
- Design Sensitivity



Langley
Research
Center
00-3

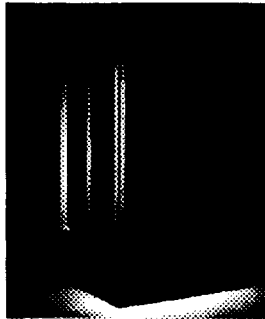
Parallel-Vector Speedup (over sequential code)



Langley
Research
Center
00-4

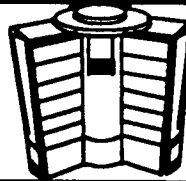
1994 Supercomputers

IBM SP-X

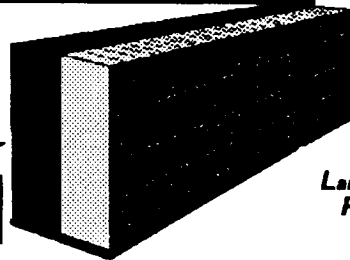


Being installed at NAS (160 proc) and LaRC (48 proc) this summer

Cray C-90 and T-3D



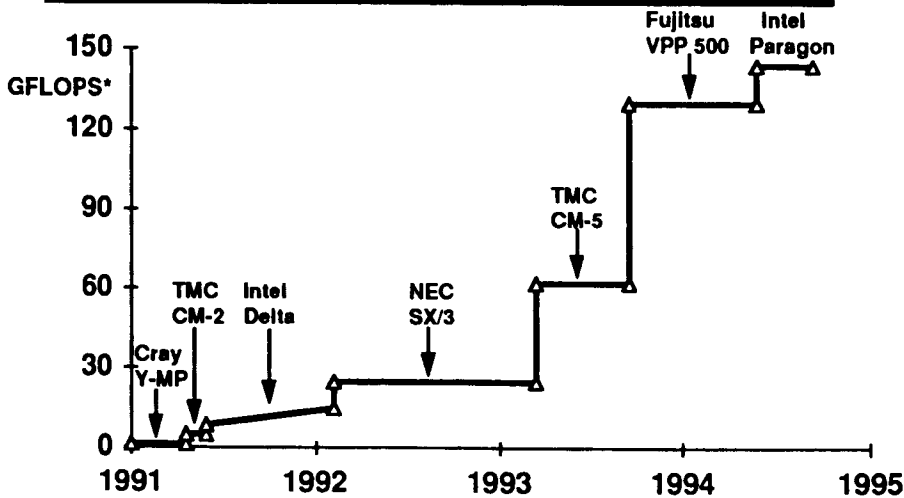
Intel Paragon



Current world record holder
143 GigaFLOPS for MP-Linpack

Langley
Research
Center

Record MP-Linpack GFLOPS*

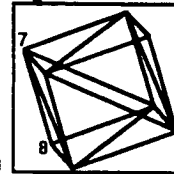
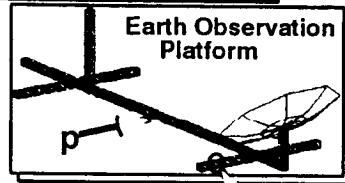


* billion floating point operations per second

Langley
Research
Center

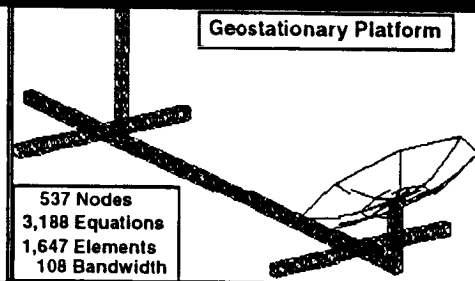
Typical Structural Analysis

- **Generate mesh**
(nodes and elements)
- **Assemble stiffness [K],**
mass [M], and load {p}
- **Solve:** $[K] \{u\} = \{p\}$ for displacement, u
 $[K] \{\phi\} = \lambda [M] \{\phi\}$ for modes, ϕ
- **Repeat:** multiple analyses for nonlinear & design
- **Plot:** u , stresses and vibration modes, ϕ

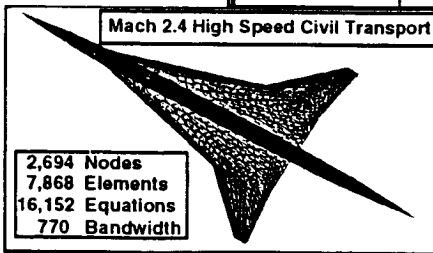


Langley
Research
Center

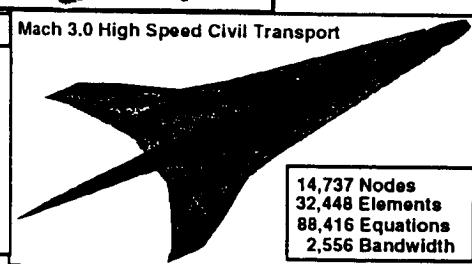
Performance Assessment Applications



537 Nodes
3,188 Equations
1,647 Elements
108 Bandwidth



2,694 Nodes
7,868 Elements
16,152 Equations
770 Bandwidth



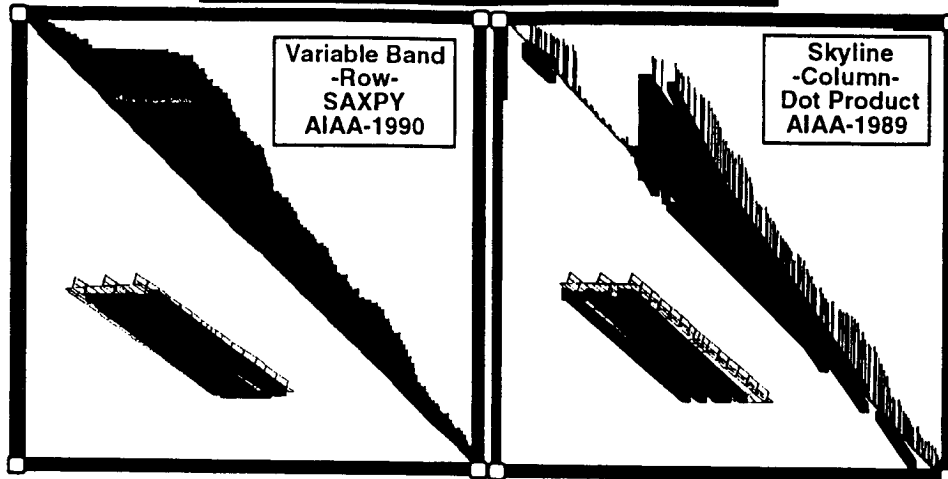
14,737 Nodes
32,448 Elements
88,416 Equations
2,556 Bandwidth



Langley
Research
Center

Matrix Storage Methods

2910 Equation Stiffened Panel



*Langley
Research
Center*
04-9

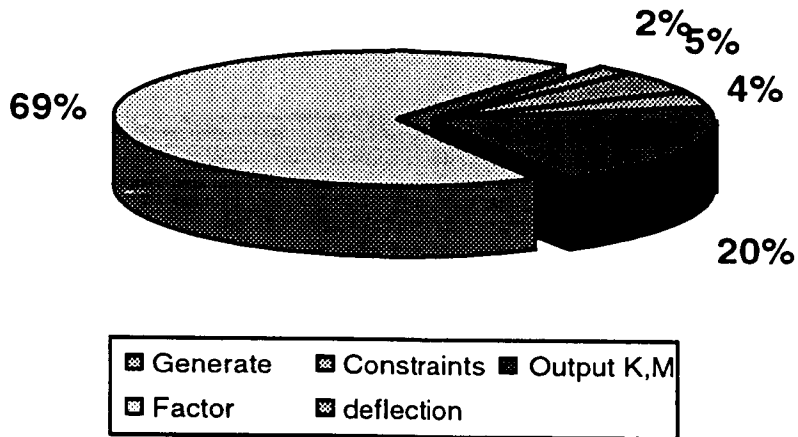
Parallel-Vector Structures Algorithms

Static	Eigenvalue	Dynamics-Control	Flutter	Optimization
$Ku = f$	$\kappa \phi = \lambda M \phi$	$M\ddot{u} + C\dot{u} + Ku = f(t)$	$K\Phi = \lambda M\Phi$	$b_{k+1} = b_k + s_k d_k$
Substructuring NL Algorithms	Subspace Lanczos	Time Integration Reduced-Order Simulate Multibody	Unsymmetric Choleski and Lanczos	Search methods Sensitivity
Matrix Assemblers - Finite Element based - Degree-of-Freedom based				
Equation Solvers - Direct - Sparse - Iterative - SVD				



*Langley
Research
Center*
04-10

Structural Analysis Computation Time

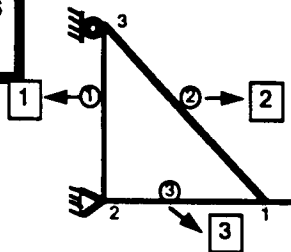


Langley
Research
Center
08-11

Parallel Matrix Generation and Assembly

By element: Traditional thinking
 Generate $[k^{(e)}]$ on different processors
 Assemble global $[K] = \sum [k^{(e)}]$

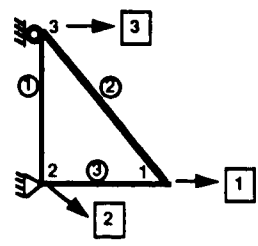
can't write elements simultaneously!



By node: New method

Nodal Connectivity

Node	Proc.	Elements
1	1	② ③
2	2	① ③
3	3	① ②



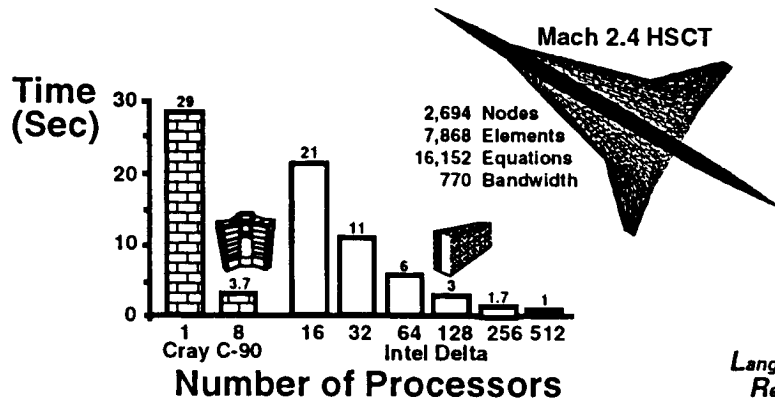
Langley
Research
Center
08-12



Parallel Structural Matrix Generator/ Assembler Demonstrated on HSCT



- Nearly ideal parallel speedup
- (no interprocessor communication)



Langley
Research
Center
08-13



Equation Solution Issues (Time, memory, disk space, I/O)



- Iterative or direct ?
- Banded or sparse ?
- "In-core" or "out-of-core" ?

Communication



- Broadcast or ring?
- OSF or SUNMOS?



Langley
Research
Center
08-14

Equation Solvers

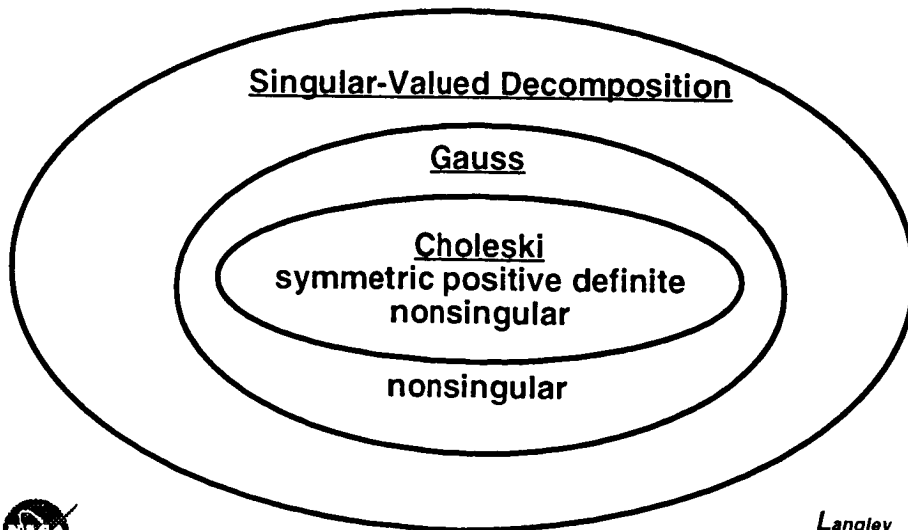
- Iterative and *Direct* (function of application)
- Linpack (MP Linpack), LAPack
(needs full matrix for best performance)
- Banded Indefinite, nonsymmetric
(requires pivoting)
- Banded Definite Symmetric
(seldom occurs in practical structures)
- Skyline*, *Variable-band**
(DOT-product, SAXPY operations minimize time)
- *Sparse**, *Wavefront** (<5% nonzeros)



* node or equation reordering minimizes solution time

Langley
Research
Center
02-15

Direct Equation Solvers



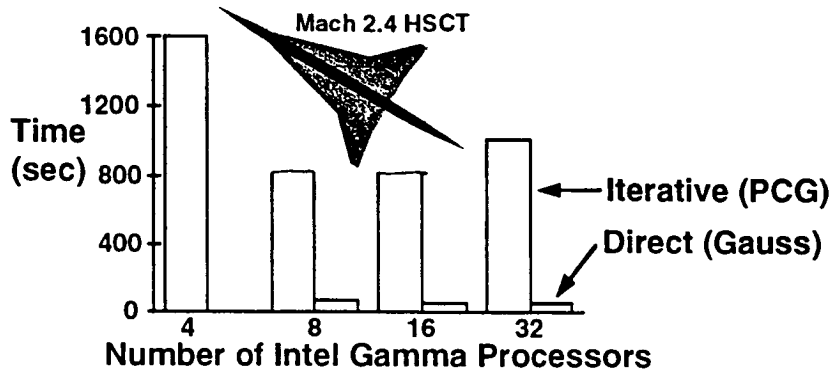
Langley
Research
Center
02-15



Iterative vs Direct Solvers



- Iterative slow, convergence not guaranteed
- Direct complex coding (banded, sparse)



Langley
Research
Center
08-17



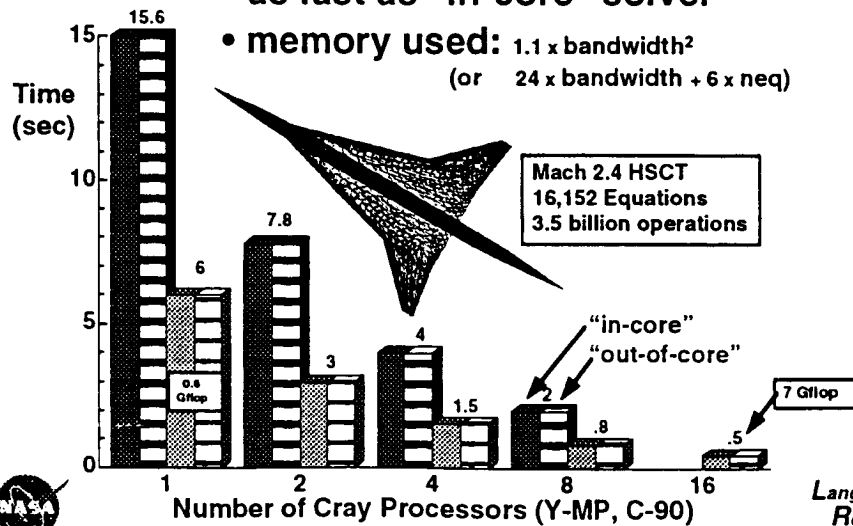
“Out-of-core” Direct Solver

- using Cray Solid State Disk -



- as fast as “in-core” solver

- memory used: $1.1 \times \text{bandwidth}^2$
(or $24 \times \text{bandwidth} + 6 \times \text{req}$)



Langley
Research
Center
08-18



Automotive Application of Sparse Solver



48,894 Elements
44,188 Nodes
263,574 Equations

- Langley solution took 40 CPU sec (1 Cray C-90 processor)
- fastest solution known to date -
- Challenge: achieve even faster solution on SP-2 and Paragon!



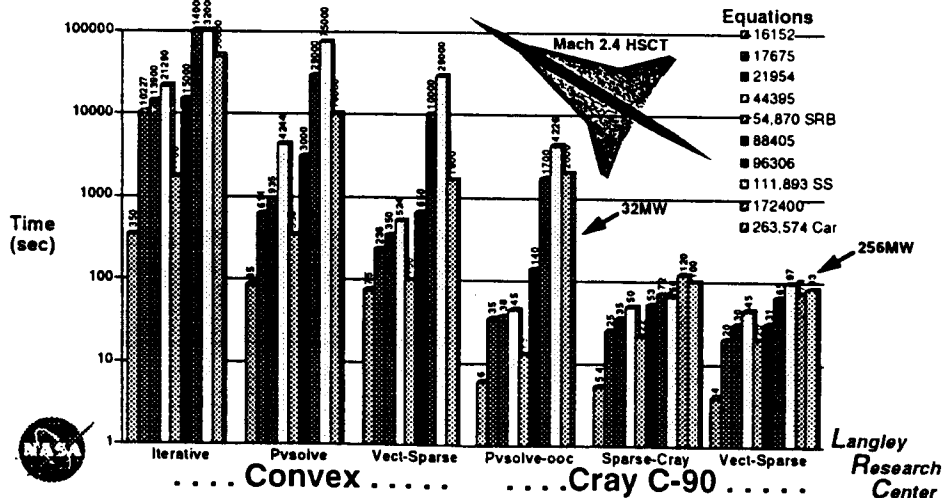
Langley
Research
Center
DS-11

Convex

Equation Solver Results - 10 Finite Element Models -



• Iterative slowest, Sparse fastest!



Langley
Research
Center
DS-11



Industrial-Strength Equation Solvers for $[A]\{x\}=\{b\}$ and $[A]\{x\}=\lambda[M]\{x\}$



Solver	Application	Memory	Parallel(shared)	Parallel(Distributed)
PVSOLVE	Symmetric + Def*	equations X Bandwidth	Yes (Cray C-90, etc)	Yes (Intel Paragon, IBM SP-1)
PVSOLVE-OOC	"	1.1 x Bandwidth ²	Yes (Cray C-90, etc)	Not yet
PVSOLVE-OOC+	"	24 x Bandwidth	No (Cray C-90, etc)	Not yet
VSS (Vector Sparse)	"	function of sparsity	No (Cray C-90, etc)	Not yet
PCG(Iterative)	"	- matrix nonzeros	Yes (Cray C-90, etc)	Yes (Intel Paragon, IBM SP-1)
LANZ(Eigensolver)	"	equations X bandwidth	Yes (Cray C-90, etc)	Yes (Intel Paragon)

NOTE: These solvers have been evaluated on real applications with up to 263,574 equations and larger matrices with several million equations. PCG is slowest, VSS is fastest (for large, sparse problems) and PVSOLVE-OOC is the best all-around parallel-vector solver. PVSOLVE-OOC exploits Cray solid-state disk.



* special versions of PVSOLVE for unsymmetric and negative coefficient matrices solve panel flutter, CFD, nonlinear and optimization problems

Langley
Research
Center
08-71



Parallel-Vector Equation Solver (PVSOLVE)



Shared Memory

Cray GigaFLOP award

- "in-core skyline and variable-band versions
- "out-of-core" versions: memory ~ 1.2 bandwidth² and 24 x bandwidth
- tuned for Crays (or shared memory computer/workstation)

Distributed Memory

- "in-core" skyline - Intel i/860 or Paragon
- "in-core" row version - Intel 860 or Paragon, IBM SP-1
- Conversion underway to TMC CM-5, Convex SSP-1 and Cray T-3D

Use



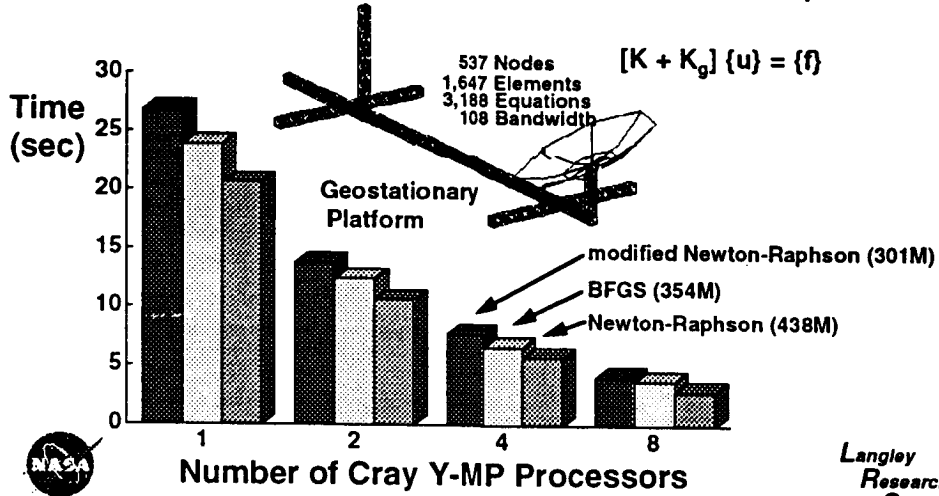
- COMET, Ford, U. Virginia, IBM, Princeton, LLNL, NSF sites
Convex, COMCO, NASA Lewis + several dozen sites

Langley
Research
Center
08-71



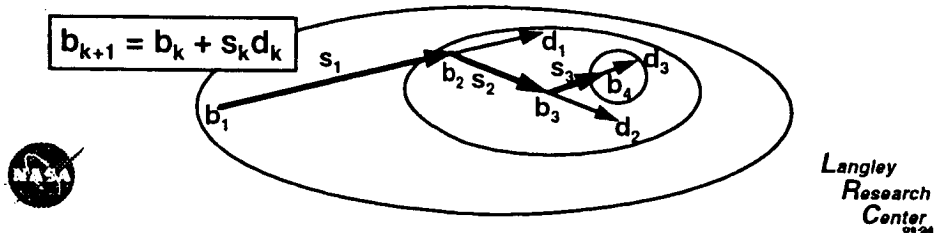
Parallel Geometric Nonlinear Methods

- Newton-Raphson fastest on parallel-vector computers



Optimization Procedure

- Find aircraft minimum weight subject to displacement and stress constraints
- Nonlinear constrained optimization finds:
 - Direction: BFGS, Simplex-Linear Programming
 - Step size: Golden Block

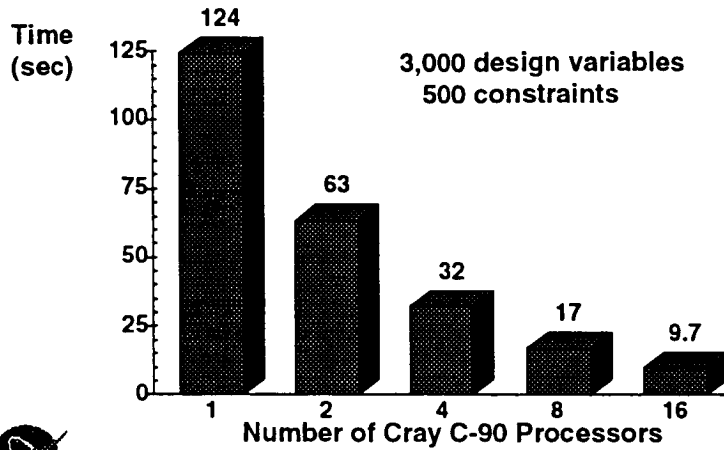




Parallel Simplex Method Linear Programming



- Scalable time reduction



Langley
Research
Center
09-25



Parallel BFGS Optimization



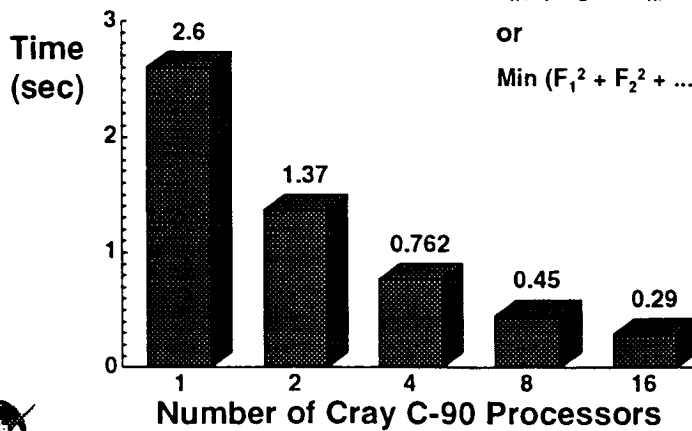
Minimize $F(x_1, x_2, \dots, x_n)$ is equivalent to

For 11,000 nonlinear equations:

$$\left. \begin{aligned} F_1(x_1, x_2, \dots, x_n) &= 0 \\ F_2(x_1, x_2, \dots, x_n) &= 0 \\ &\vdots \\ F_n(x_1, x_2, \dots, x_n) &= 0 \end{aligned} \right\}$$

or

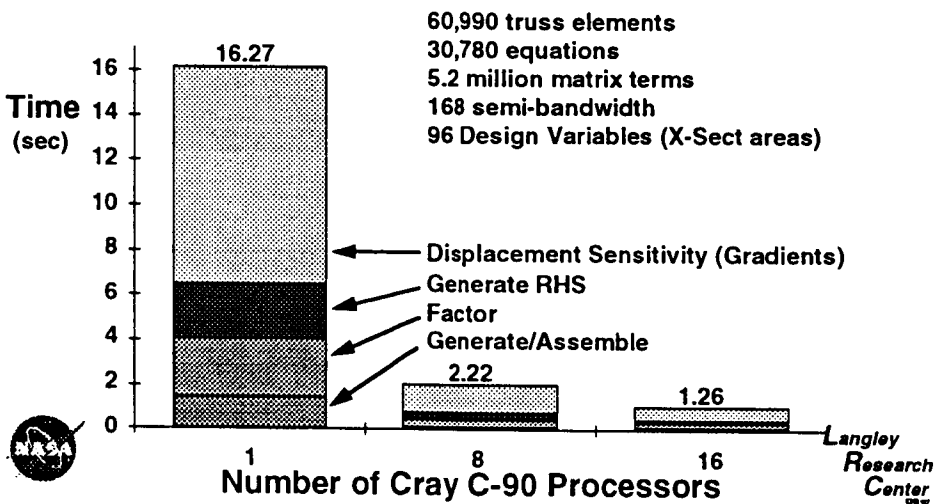
$$\text{Min } (F_1^2 + F_2^2 + \dots + F_n^2)$$



Langley
Research
Center
09-26

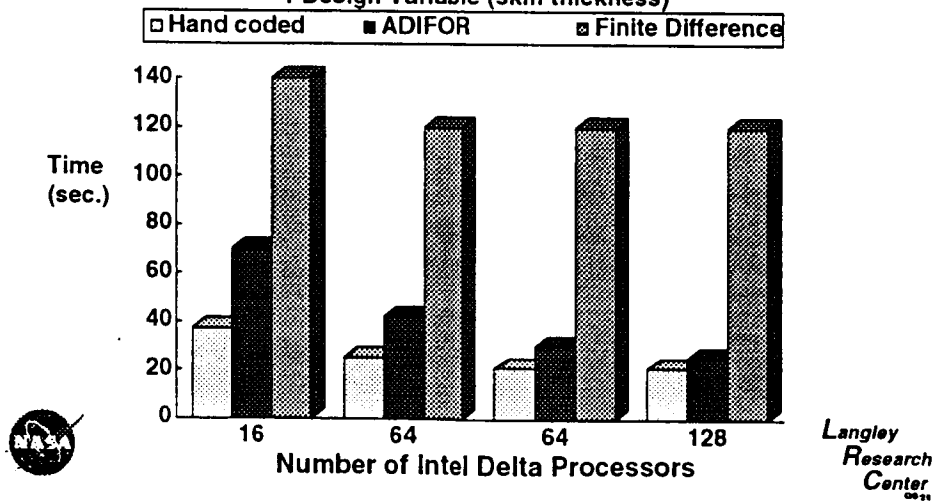
Displacement Sensitivity Analysis by Automatic Differentiation (ADIFOR)

– 2-D Truss (80 bays x 190 stories)



Design Sensitivity Analysis Methods for Mach 2.4 HSCT

7,868 Triangular Elements
1 Design Variable (skin thickness)





Concluding Remarks



- New algorithms for high-performance computers
- Perform well on large-scale applications:
 - Nodal Matrix Generation and Assembly
 - Equation Solvers: $[K]\{u\} = \{p\}$
(linear, nonlinear, "out-of core", sparse)
- Structural Optimization
 - Design Sensitivity
- *Operate on Cray, Paragon, IBM SP-1 and SP-2!*



Langley
Research
Center
08-28

References

- Storaasli, O., Nguyen, D., Baddourah, M. and Qin, J.;
"Computational Mechanics Analysis Tools for Parallel-
Vector Supercomputers", *AIAA/ASME/ASCE/AHS/ASC
34th Structures, Structural Dynamics and Materials
Conference Proceedings, Part 2*, pp. 772-778, April 1993.
- also *International Journal of Computing Systems in
Engineering*, Vol. 4, No. 2-4, 1993 pp. 349-354
- on MOSAIC-WWW (Langley Technical Report Server)
- Questions: O.O.Storaasli@larc.nasa.gov
- Free Videotape from: shuguez@nas.nasa.gov
(Santa Huguez at 415-604-4632)



Langley
Research
Center
08-30

Solution of Matrix Equations Using Sparse Techniques

P. 9

by Majdi Baddourah, (majdi@sunny.larc.nasa.gov or 804-864-2913)

The solution of large systems of matrix equations is key to the solution a large number of scientific and engineering problems.

Tradition has it that iterative methods persist for CFD and direct methods for Structures applications. With the increase in computational power (over 3 orders of magnitude this decade) problem sizes with full detail that could not have even been considered tractable are now solved routinely. The equation solvers used for structures applications have advanced from the use of full matrix (LINPACK, LAPACK BLAS-3) to band solvers to variable band and skyline solvers to sparse matrix solvers with corresponding increases in performance. It appears that for large-scale structural analysis applications sparse matrix methods have a significant performance advantage over other methods. This talk will describe the latest sparse matrix solver developed at Langley which if not the fastest in the world is among the best. It can routinely solve in excess of 263,000 equations in 40 seconds on one Cray C-90 processor.

=====
Dr. Majdi Baddourah received the Ph D. in the Department of Civil Engineering at Old Dominion University in 1991. He has been employed by Lockheed Engineering and Sciences Company since then in support of the Computational Structures Branch at NASA Langley Research Center. Dr. Baddourah is widely recognized for contributing to the development of software to exploit scalable high-performance computers for structural analysis applications including the solution of large systems of equations (approaching 1 million) by both direct and iterative methods.

Solution of Matrix Equations Using Sparse Techniques

**Majdi A. Baddourah
Lockheed Engineering and Sciences Co.**

**1994 Workshop
June 15-16**

Outline

- **Matrix Storage**
- **Reordering**
- **Factoring**
- **Results (Computational Structures and Fluids)**
- **Conclusion**

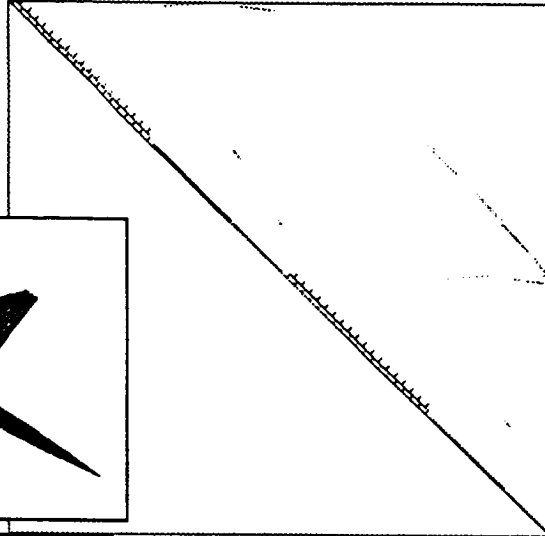
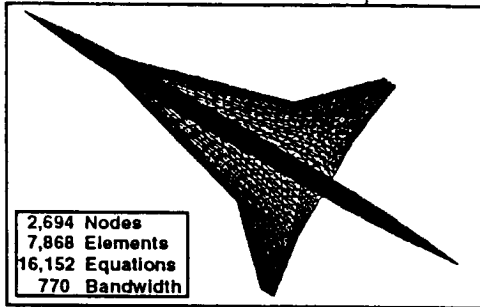
Original Matrix (No reordering)

DOF

1x
1y
1z
1rx
1ry
1rz

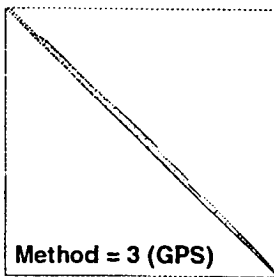
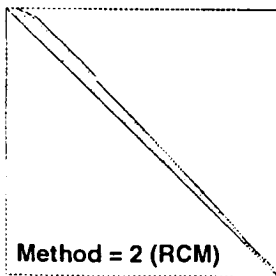
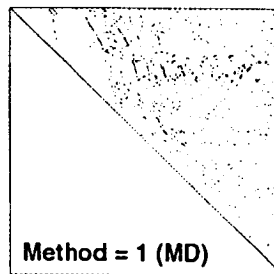
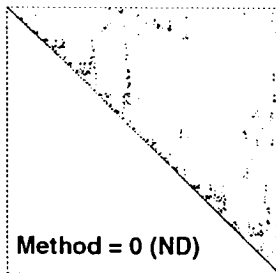


1 Node = 6 DOF = 6 Equations

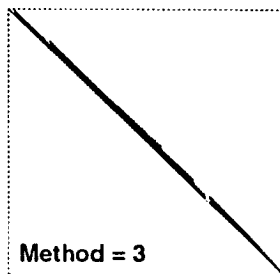
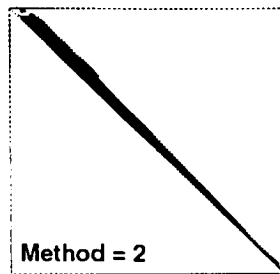
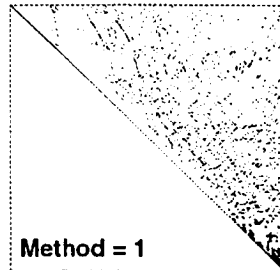
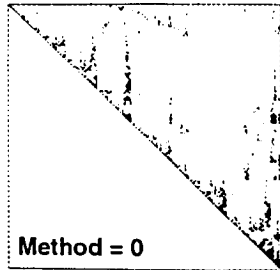


Mach 2.4 High Speed Civil Transport

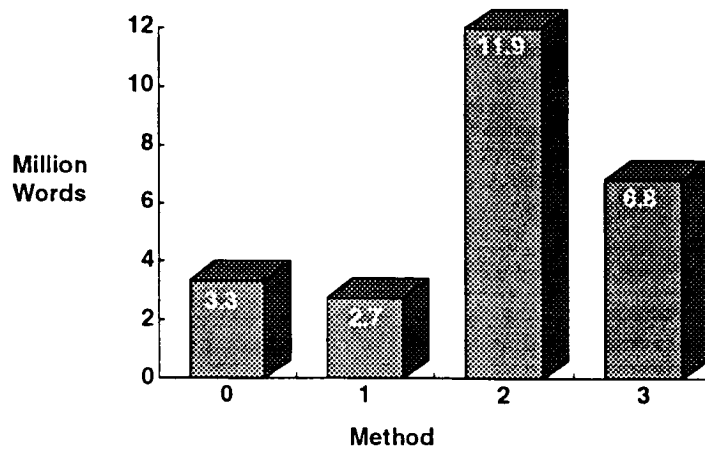
Reordering Methods



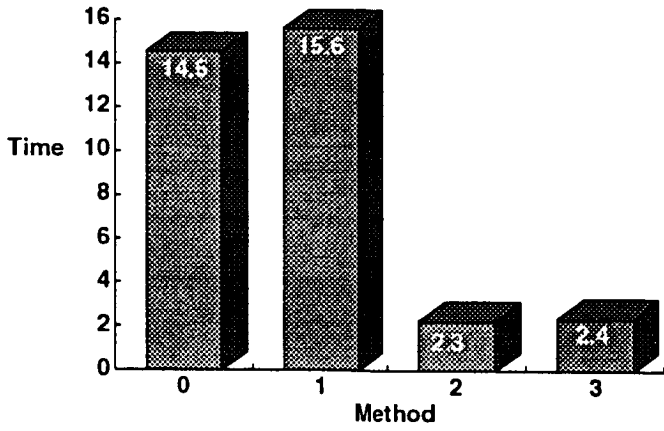
Matrix After Factoring



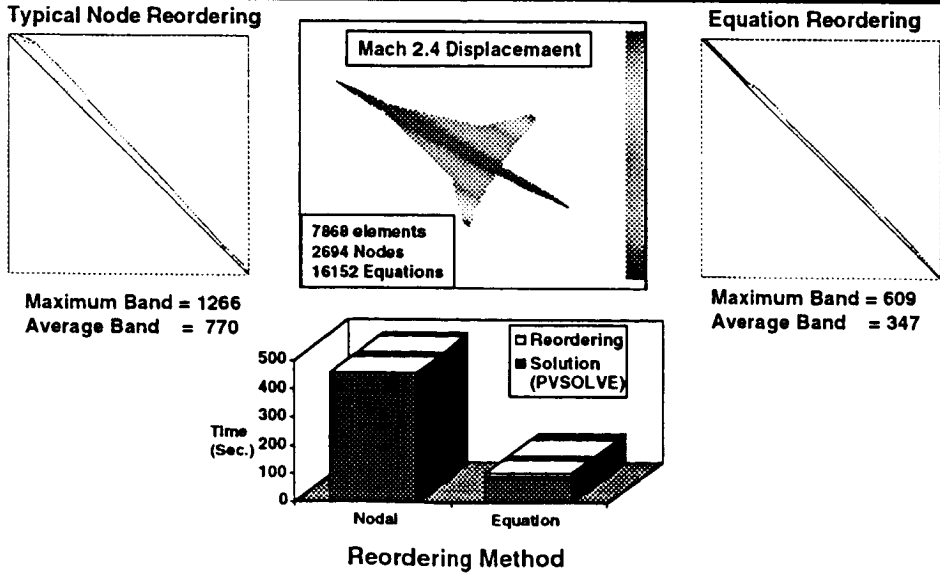
Matrix Storage Memory Requirement



Reordering Time



Equation Reordering Reduces Solution Time



Factoring Matrix

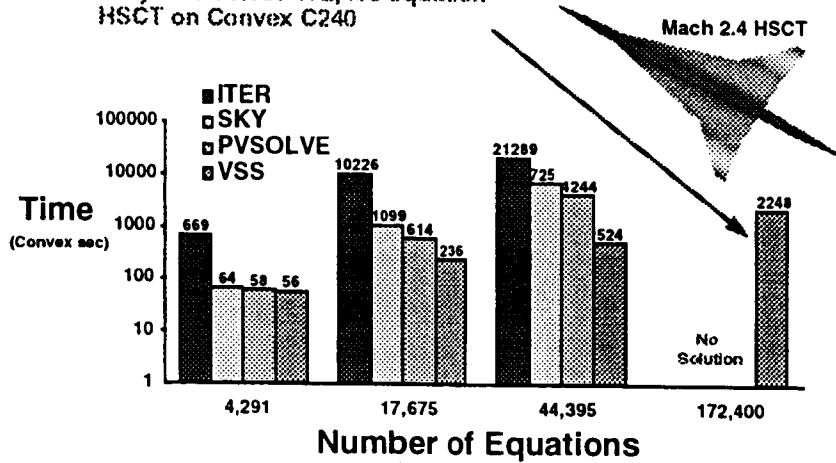
- **Banded or full:**
 - easy to vectorize.
 - problem size limit.
- **General sparse:**
 - difficult to vectorize.
 - fewer operations.
 - indirect addressing.

Results

- High Speed Civil Transport
- Space Station
- CFD Application
- Automotive Application

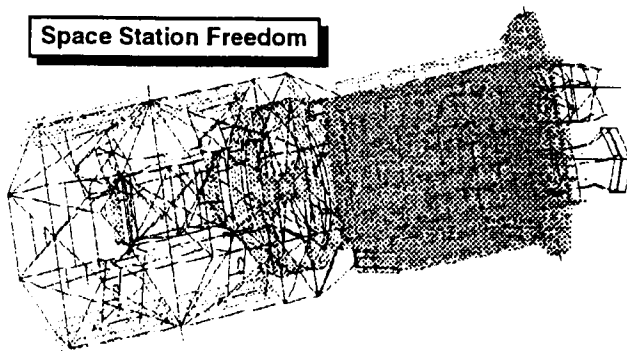
Mach 2.4 HSCT Results

- Only VSS solves 172,400 equation HSCT on Convex C240



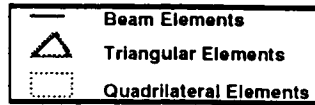
Space Station Application

Space Station Freedom



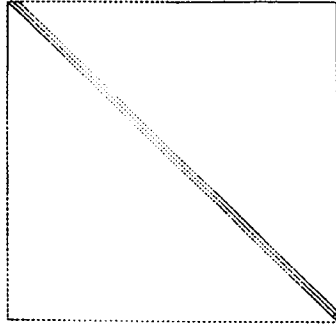
111893 Equations
1664984 Non-zero terms
97 solution secs*

* Using 1 Cray Y-MP processor and Solid State Disk at NAS

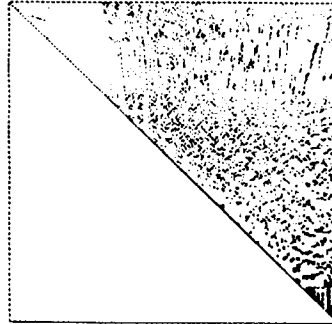


CFD Application

Before Reordering



After Reordering with fill



Number of Equations = 15360
Number of Coefficients = 257797

Number of Coefficients = 3081995

1 Cray C-90 Solution Time = 6.7 Seconds

Automotive Application



44,188 Nodes
48,894 Elements
263,574 Equations

NASA solution took 78 sec for full static analysis
(on 1 Cray C-90 processor)

- fastest solver known to date -

(32 sec reordering, 45 sec factor and 1 sec F/B)

CRAY Sparse solver took 102 sec for full static analysis
Banded Solver took 2500 sec for full static analysis

Conclusion

- **Sparse solvers are preferred for large-scale structures.**
- **Sparse Solver outperforms iterative solver which can have convergence problems.**
- **Sparse Solver can be used for CFD applications**
- **Sparse solvers use minimum memory.**

Equation Solvers for Distributed Memory Computers^{P. 12}

by Olaf O. Storaasli, (O.O.Storaasli@larc.nasa.gov or 804-864-2927)

for Workshop on the Role of Computers in Langley R&D (6-15-94)

A large number of scientific and engineering problems reduce to the solution of large systems of simultaneous equations. Solving large systems of simultaneous equations rapidly thus makes the solution of large-scale structures, physics, electromagnetics and fluid mechanics problems tractable. The performance of parallel computers now dwarfs traditional vector computers by nearly an order of magnitude, so the challenge is to rapidly solve large systems of equations rapidly on the new breed of scalable parallel processing supercomputers.

Research at Langley on solving equations on distributed memory computers goes back nearly ten years to the Langley Finite Element Machine, one of the nation's first parallel computers with 32 processors developed by NASA before commercial parallel computers were available. Since then, both iterative and direct parallel equation solvers have been developed and tuned for parallel computers manufactured by Flexible computer, N-Cube, Alliant, Encore, Cray, Intel, Convex and IBM. The solvers, PVSOLVE and PVS-MP are currently running on the IBM SP-1 and SP-2 under a Memorandum of Agreement with IBM which permits Langley early access to the SP-1 and SP-2 in return for IBM given permission to use the NASA solvers for advertisements, demonstrations, and trade shows. These Langley solvers are timely since in a recent \$22.4 million procurement, two IBM SP-2 supercomputers will be delivered to NASA (160 processors to NAS and 48 processors to LaRC). Based on benchmarks and the Langley parallel equation solvers, these IBM supercomputers promise to surpass the performance of traditional Cray vector supercomputers and other parallel computers.

The talk will describe the major issues involved in parallel equation solvers with particular emphasis on implementations the Intel Paragon, IBM SP-1 and SP-2.

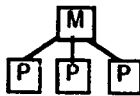


Equation Solvers for Distributed-Memory Computers

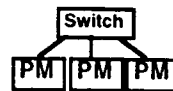


Dr. Olaf Storaasli
Computational Structures Branch
Mail Stop 240
NASA Langley Research Center
Hampton, VA 23681

Email: O.O.Storaasli@larc.nasa.gov
Phone: 804-864-2927
FAX: 804-864-8912



presented at



Workshop on
The Role of Computers in Langley R&D
June 15, 1994, Reid Auditorium
Langley Research Center



*Langley
Research
Center*

Objective

- ***Faster, cheaper, better* analysis/design of large-scale structures**
 - Develop algorithms to exploit distributed-memory computers
 - Evaluate computational performance



*Langley
Research
Center*



Outline



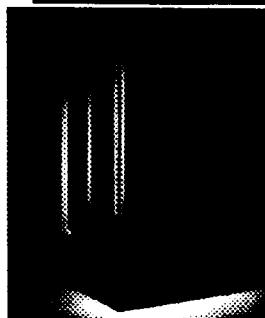
- Distributed-memory Computers
- Structural Applications
- Structural Analysis
 - - Nodal Generation and Assembly
 - - Linear Equation Solvers
- Structural Optimization
- X-Design Sensitivity



Langley
Research
Center
00-3

1994 Distributed-Memory Supercomputers

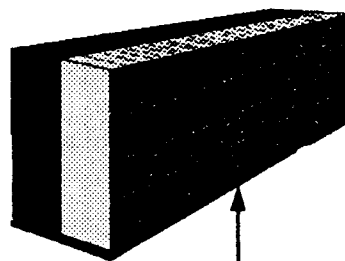
IBM SP-2



Being installed this summer at
NAS (160 proc)
and LaRC (48 proc)
266 MFLOPS/proc peak



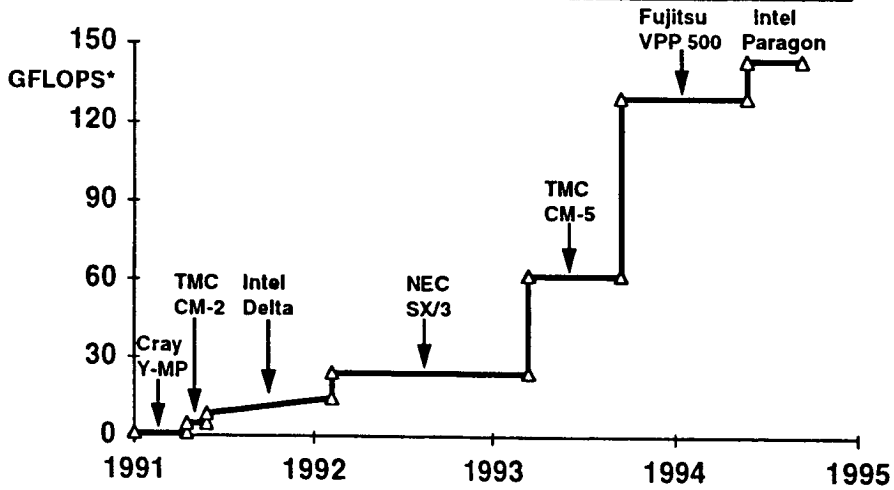
Intel Paragon



Current world record holder!
143 GigaFLOPS for MP-Linpack

Langley
Research
Center
00-1

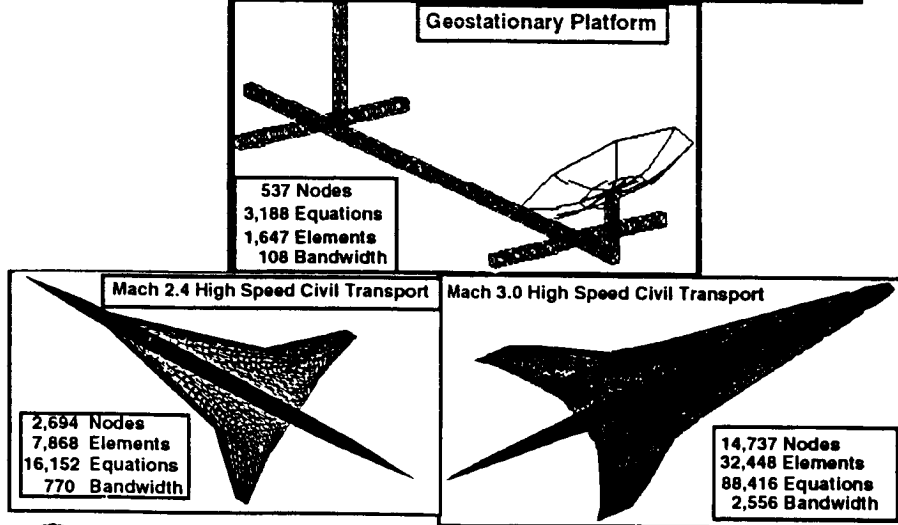
Record MP-Linpack GFLOPS*



* billion floating point operations per second

Langley
Research
Center

Performance Assessment Applications

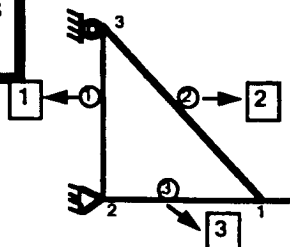


Langley
Research
Center

Parallel Matrix Generation and Assembly

By element: Traditional thinking
 Generate $[k^{(e)}]$ on different processors
 Assemble global $[K] = \sum [k^{(e)}]$

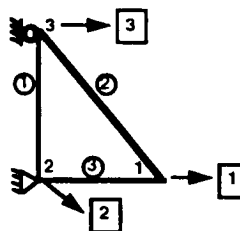
can't write elements simultaneously!



By node: New method

Nodal Connectivity

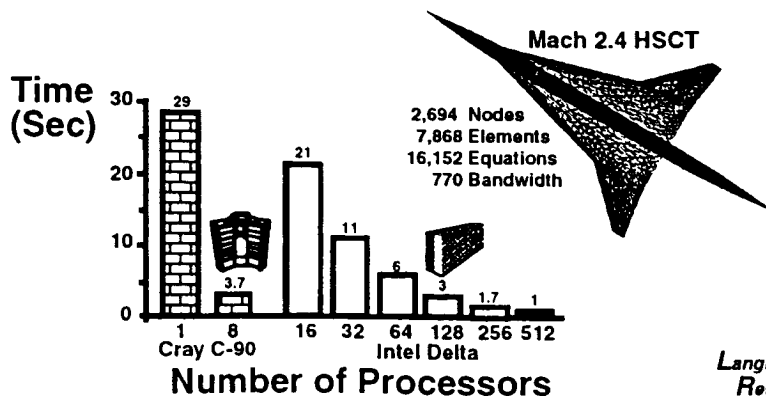
Node	Proc.	Elements
1	1	(2) (3)
2	2	(1) (3)
3	3	(1) (2)



Langley
 Research
 Center

Parallel Structural Matrix Generator/ Assembler Demonstrated on HSCT

- Nearly ideal parallel speedup
 (no interprocessor communication)



Langley
 Research
 Center

Equation Solution Issues

(Time, memory, disk space, I/O)

- Iterative or direct ?
- Banded or sparse ?
- “In-core” or “out-of-core” ?

Communication

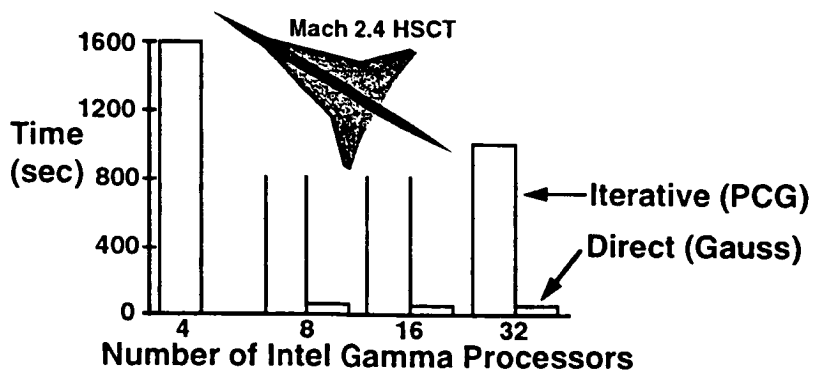
- Broadcast or ring?
- OSF or SUNMOS?



Langley
Research
Center
00-1

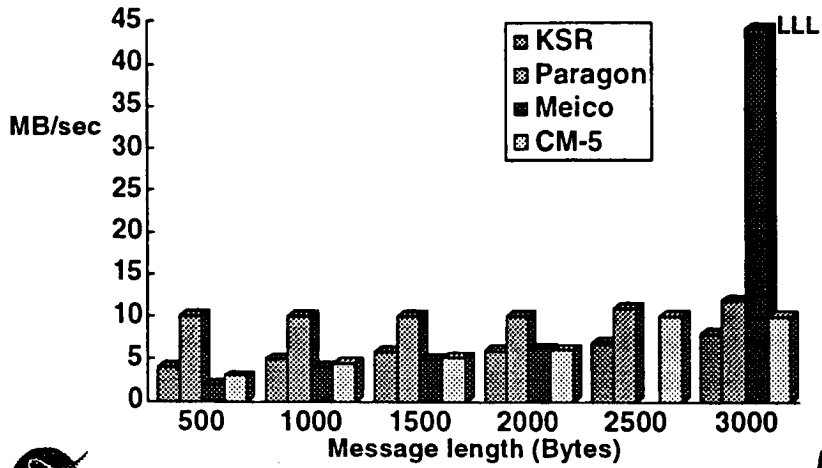
Iterative vs Direct Solvers

- Iterative slow, convergence not guaranteed
- Direct complex coding (banded, sparse)



Langley
Research
Center
00-10

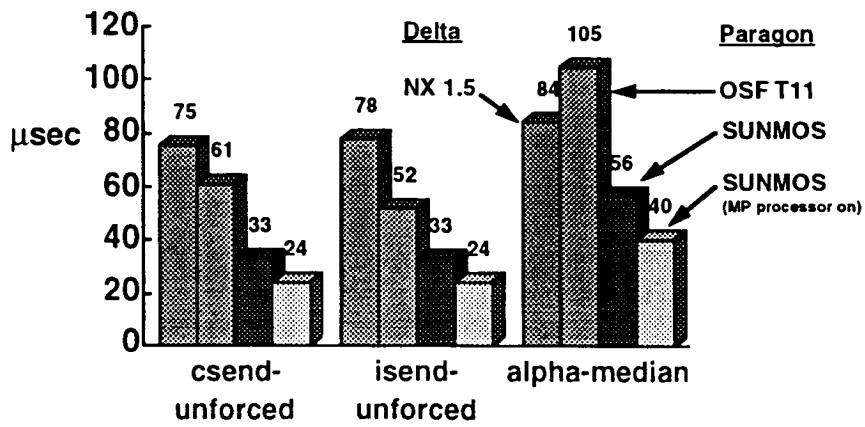
Interprocessor Communication



Langley
Research
Center
DS-11



Latency OSF vs SUNMOS



Langley
Research
Center
DS-11



Paragon Status



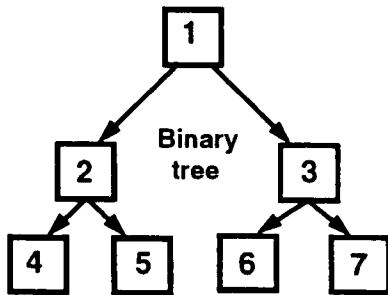
- **OSF Rev 1.1** (Latency: 150 -> 85 μ sec, Tools Communication: 11 -> 34 MB/sec, Memory 8 -> 6MB)
- **OSF Rev 1.2** (Latency: 85 -> 50 μ sec, Communication: 34 -> 55 MB/sec)
- **New comm chip:** tested at 400 MB/sec
- **Dynamic Memory:** avoid inconsistencies (i.e. faster 2nd runs)
- **SUNMOS: Sandia-UNM O/S**
(Latency: 24 μ sec, Comm: 175 MB/sec, Mem: 0.3MB)
178 MB/sec on Grace (NAS benchmarks run faster)



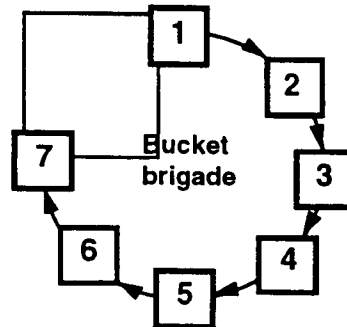
Langley
Research
Center
88-13

Interprocessor Communication Methods

Broadcast
(widely used)



Ring



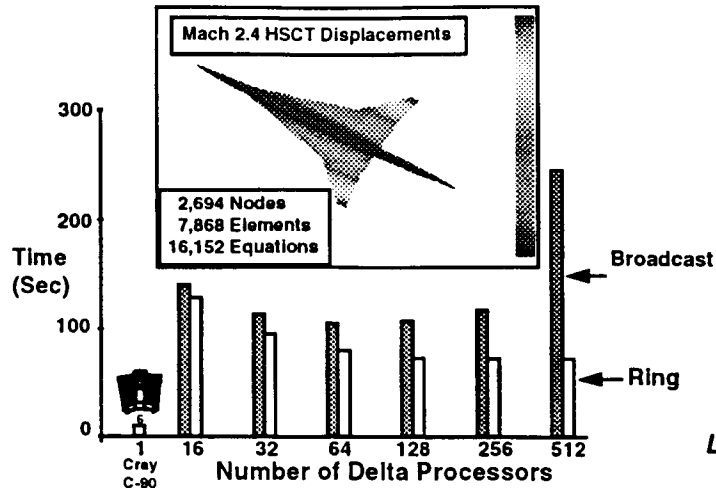
Langley
Research
Center
88-14



Solution Time: Mach 2.4 HSCT



- Ring communication reduced solution time
- Slower than 1 Cray processor



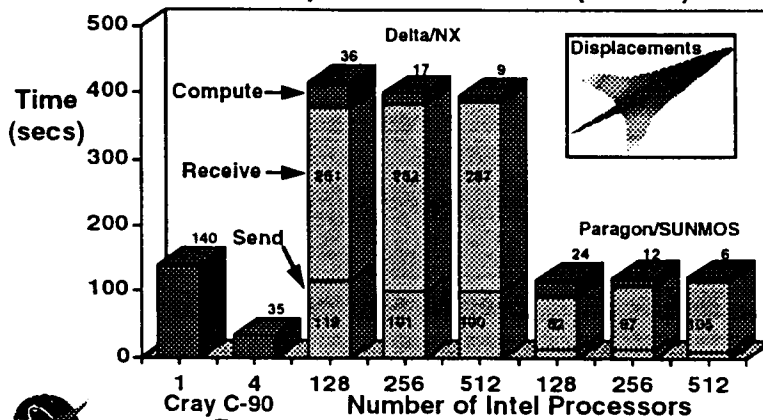
Langley
Research
Center
03-15



Solution Time Breakdown - Mach 3.0 HSCT -

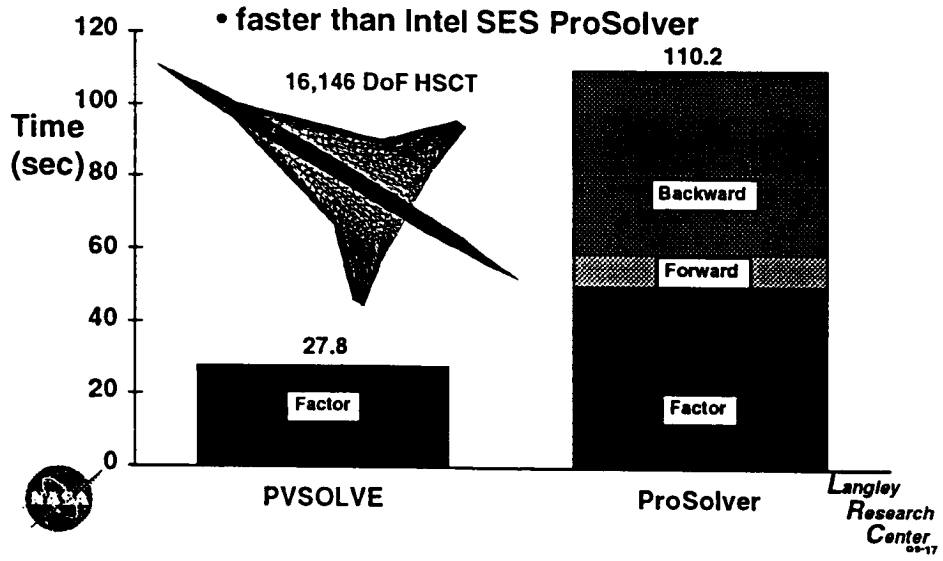


- Communication dominates
- Computation scalable (< C-90)

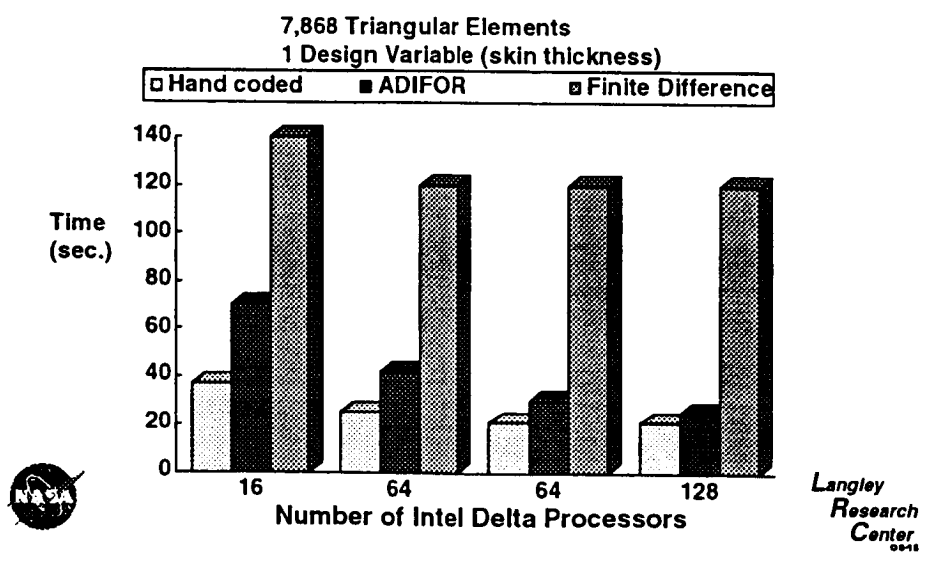


Langley
Research
Center
03-16

Solver Performance - 32 Gamma Processors -



Design Sensitivity Analysis Methods for Mach 2.4 HSCT





Concluding Remarks



- New algorithms for distributed-memory computers
- Perform well on large-scale applications:
 - Nodal Matrix Generation and Assembly
 - Equation Solvers: $[K]\{u\} = \{p\}$
(linear, nonlinear, "out-of core", sparse)
- Structural Optimization
 - Design Sensitivity
- Operate on Paragon, IBM SP-1 and SP-2!



Langley
Research
Center
01-10

References

- Storaasli, O., Nguyen, D., Baddourah, M. and Qin, J.; Computational Mechanics Analysis Tools for Parallel-Vector Supercomputers", *AIAA/ASME/ASCE/AHS/ASC 34th Structures, Structural Dynamics and Materials Conference Proceedings, Part 2*, pp. 772-778, April 1993.
- also *International Journal of Computing Systems in Engineering*, Vol. 4, No. 2-4, 1993 pp. 349-354
- on MOSAIC-WWW (Langley Technical Report Server)
- Questions: O.O.Storaasli@larc.nasa.gov
- Free Videotape from: shuguez@nas.nasa.gov
(Santa Huguez at 415-604-4632)



Langley
Research
Center
01-10

SESSION 5 Automatic Differentiation

Chaired by

Olaf Storaasli

- 5.1 Applications of Automatic Differentiation in Computational Fluid Dynamics - Larry Green
- 5.2 Automatic Differentiation for Design Sensitivity Analysis of Structural Systems Using Multiple Processors - Duc Nguyen, Olaf Storaasli, Jiangning Qin and Ramzi Qamar

356064

110046

N95-16461

**Applications of Automatic Differentiation
in Computational Fluid Dynamics**

P. 13

Lawrence L. Green, Perry A. Newman, and Kara J. Haigler
Multidisciplinary Design Optimization Branch
Fluid Mechanics and Acoustics Division

Automatic differentiation (AD) is a powerful computational method that provides a means for computing exact sensitivity derivatives (SD) from existing computer programs for use in multidisciplinary design optimization (MDO) or in sensitivity analysis. The Mathematics and Computer Sciences Division of Argonne National Laboratory and the Center for Research on Parallel Computation at Rice University have developed a pre-compiler AD tool for FORTRAN programs called ADIFOR. The ADIFOR tool has been easily and quickly applied by NASA Langley researchers to assess the feasibility and computational impact of AD in MDO with several different FORTRAN programs. These include a state-of-the-art three-dimensional multigrid Navier-Stokes flow solver for wings or aircraft configurations in transonic turbulent flow. With ADIFOR, the user specifies sets of independent and dependent variables within an existing computer code. ADIFOR then traces the dependency path throughout the code, applies the chain rule to formulate derivative expressions, and generates new code to compute the required SD matrix. The resulting ADIFOR-generated codes have been verified to compute exact nongeometric and geometric SD, for a variety of cases, in less time than is required to compute the SD matrix using centered divided differences.

APPLICATIONS OF AUTOMATIC DIFFERENTIATION IN COMPUTATIONAL FLUID DYNAMICS

L. Green*, A. Carle**, C. Bischof***,
K. Haigler*, and P. Newman*

*NASA Langley Research Center

**Rice University

***Argonne National Laboratory

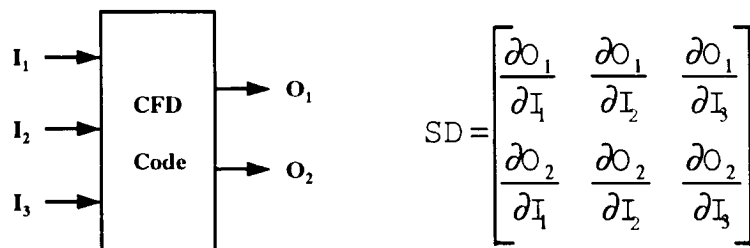
What are Sensitivity Derivatives?

- ◆ Sensitivity Derivatives (SD) describe how one thing changes with respect to another thing
- ◆ Example:
 - How a car's speed changes when braking
 - slowly at first, then more quickly
(how much)
 - speed decreases as braking increases
(which way)
- ◆ SD's describe how much and which way to change the variables in a multidisciplinary design optimization (MDO)

Objectives

- ◆ Obtain exact SD using the computational technique of Automatic Differentiation (AD)
- ◆ Assess the feasibility and computational impact of AD in a typical MDQ problem

The SD Matrix



- ◆ Sample inputs: Mach number or geometry
- ◆ Sample outputs: wing pressure coefficients or grid
- ◆ SD matrix required in MDQ

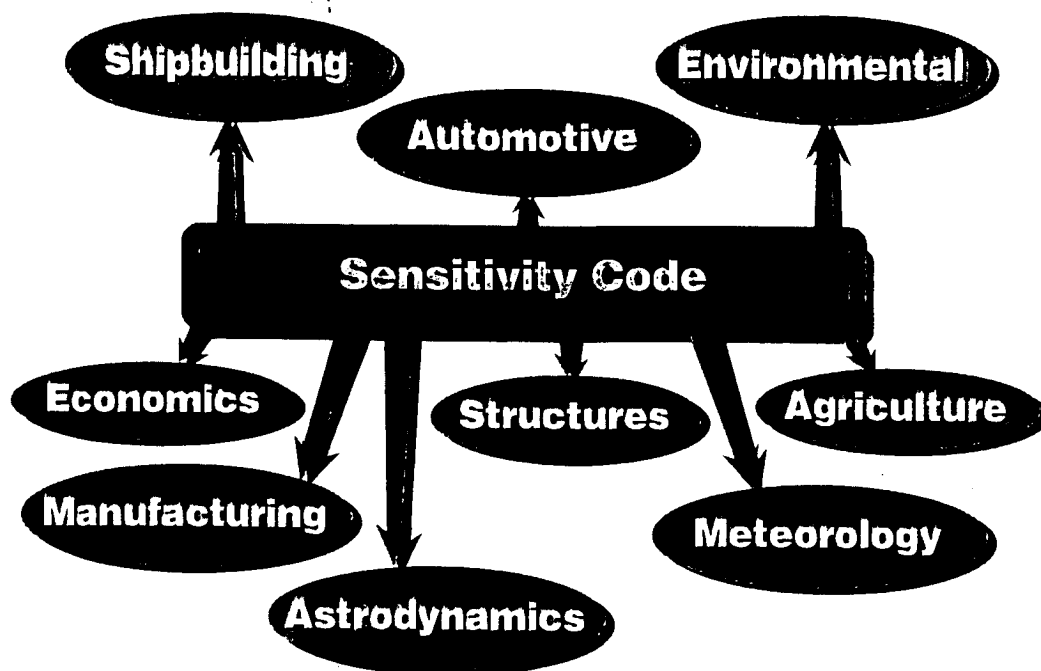
Calculation of the SD Matrix

- ◆ Divided differences (DD) (baseline + perturbations)
 - Proper step size difficult to determine
 - Truncation & resolution errors
- ◆ Hand coding (quasi-analytical) / symbolic manipulators
 - Manual dependency checking
 - Error prone and time consuming
- ◆ Automatic Differentiation (AD)
 - Automatic dependency checking and derivative coding
 - Exact derivatives via chain rule
 - Quick & easy; possible speed-up over DD

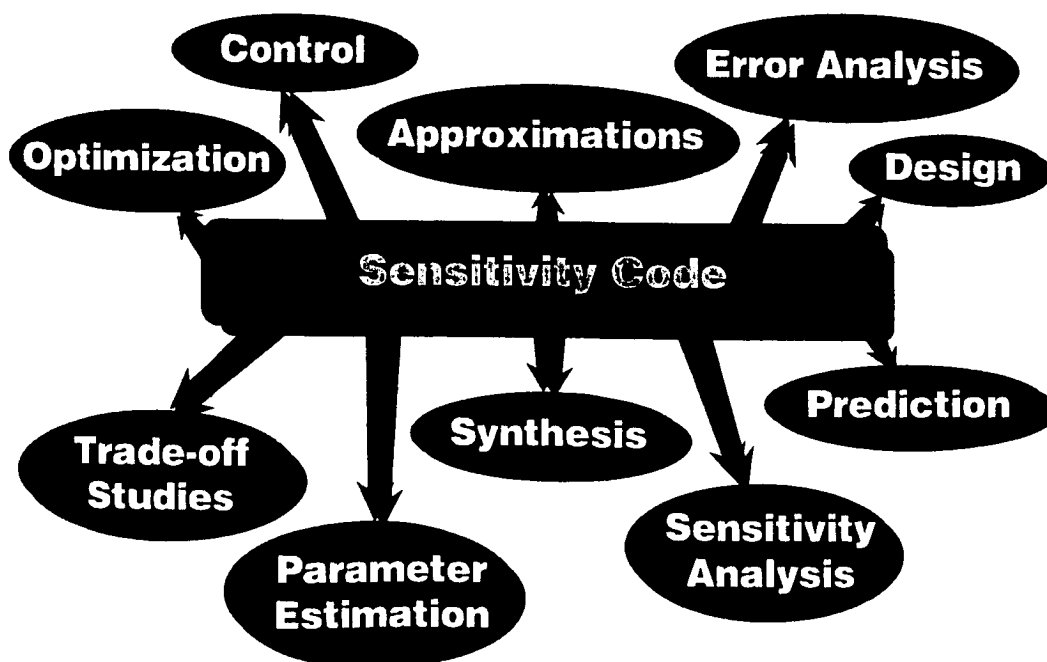
The AD Tool

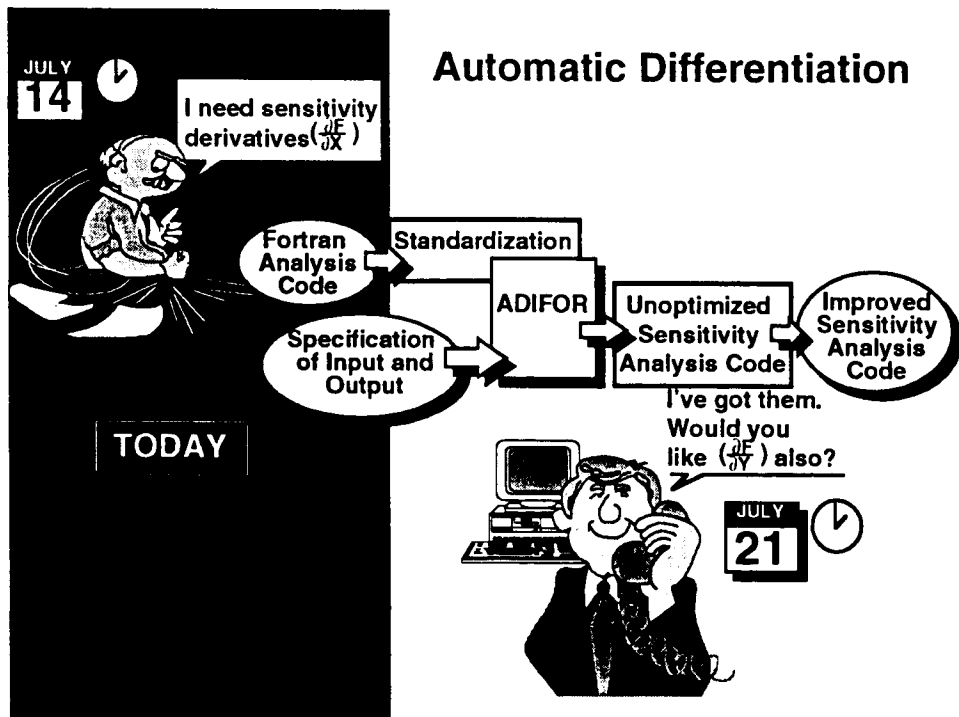
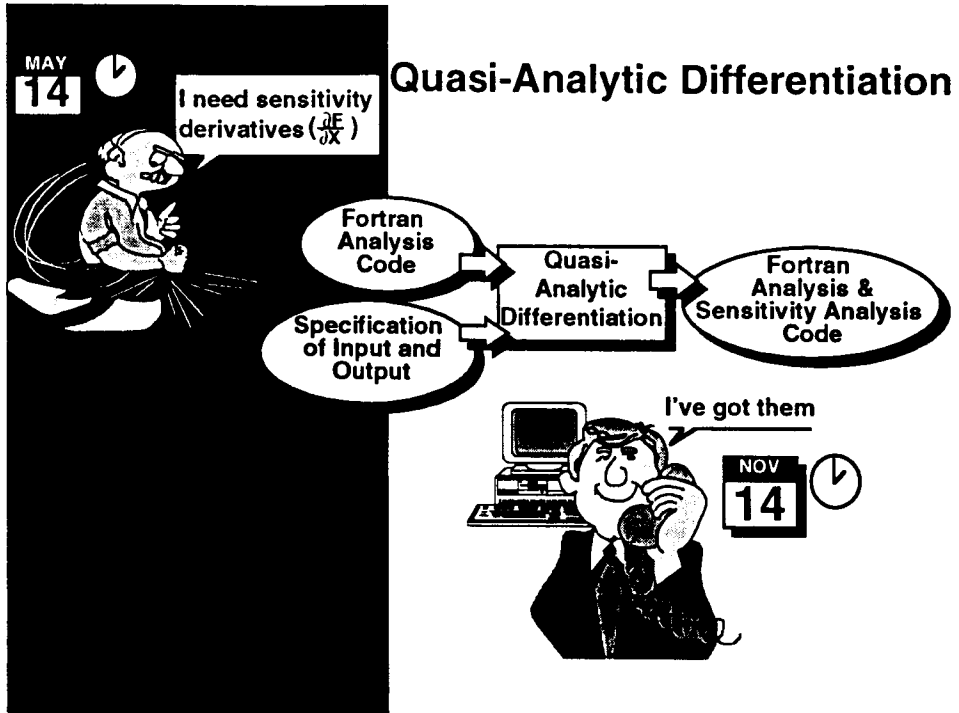
- ◆ ADIFOR (AD of FORTRAN) / PARASCOPE (Argonne National Laboratory and Rice University)
- ◆ User identifies dependent and independent variables in program
- ◆ ADIFOR follows program flow, traces program dependency paths
- ◆ ADIFOR formulates exact derivatives via the chain rule
- ◆ ADIFOR generates new code for derivative objects

Potential ADIFOR Use: By Application

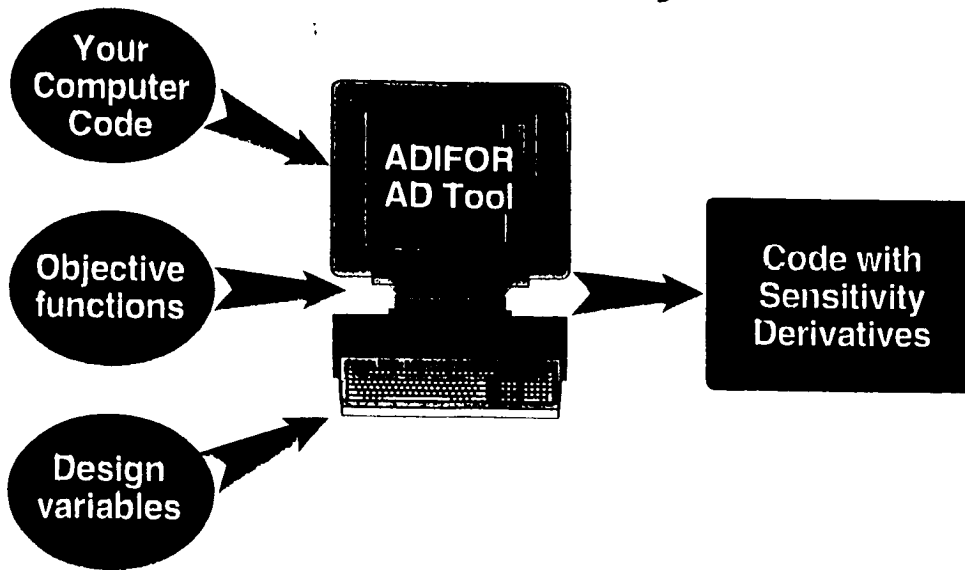


Potential ADIFOR Use: By Problem Type



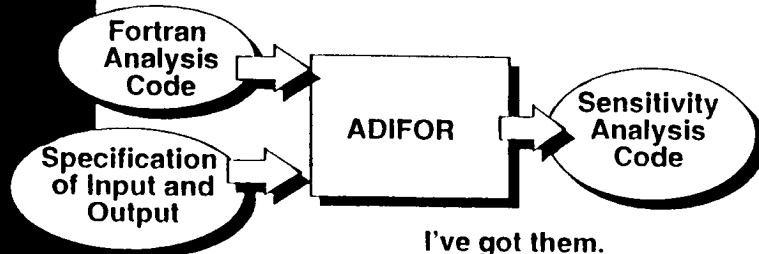


Building a Sensitivity Code



A vertical panel representing an ideal scenario. At the top left, a calendar shows "JULY 14" with a downward arrow. A speech bubble from a character says "I need sensitivity derivatives ($\frac{\partial F}{\partial X}$)". Below the character is a box labeled "IDEAL".

Automatic Differentiation



A character on a telephone. A speech bubble says "I've got them. Would you like ($\frac{\partial G}{\partial Y}$) also?". To the right, a calendar shows "JULY 14" with a leftward arrow.

The CFD Codes

- ◆ WTCO: wing C-O grid generation
 - Algebraic
 - Transfinite interpolation
- ◆ TLNS3D: 3-D thin-layer Navier-Stokes solver
 - Finite-volume, central-differencing
 - Grid sequencing, multigrid
 - Scalar artificial dissipation
 - Baldwin-Lomax turbulence model

ADIFOR Applications in CFD

- ◆ WTCO wing grid generation program
 - Independents: thickness, c_{max} , twist
 - Dependents: grid coordinates (x, y, z)
- ◆ TLNS3D Navier-Stokes flow solver
 - Independents: grid coordinates (x, y, z)
 - Dependents: pressure coefficients (C_p)

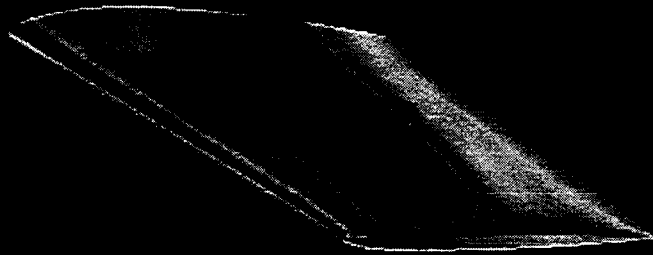
ADIFOR Applications in CFD

- ◆ WTCO wing grid generation program
 - Independents: thickness, cmax, twist
 - Dependents: grid coordinates (x, y, z)
- ◆ TLNS3D Navier-Stokes flow solver
 - Independents: grid coordinates (x, y, z)
 - Dependents: pressure coefficients (C_p)
- ◆ WTCO-TLNS3D coupling via file transfer
 - Grid
 - Grid SD matrix
 - Application of chain rule $\frac{\partial(\text{Flow})}{\partial(\text{Sect})} = \frac{\partial(\text{Flow})}{\partial(\text{Grid})} \frac{\partial(\text{Grid})}{\partial(\text{Sect})}$

Computational Results

- ◆ ONERA M6 wing planform
- ◆ NACA 2412 airfoil sections
- ◆ $97 \times 25 \times 17$ grid
- ◆ $M_\infty = 0.84$, $\alpha = 0.00$, $Re = 11.7 \times 10^6$
- ◆ Wing C_p and SD of wing C_p
- ◆ Coloring: white/red = large, blue/black = small
- ◆ Several geometries:
 - baseline
 - thickness perturbations \pm
 - cmax perturbations \pm
 - twist perturbations \pm

Flow over a wing, $M=0.84$



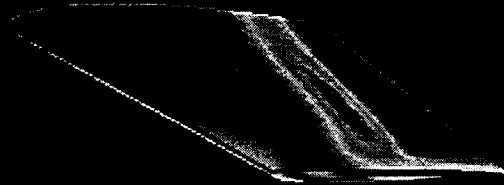
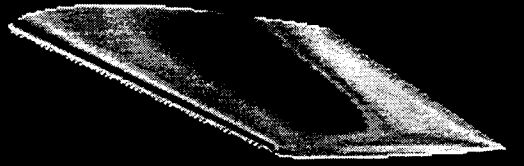
◇ Pressure coefficient (C_p)

SD for wing via AD, $M=0.84$



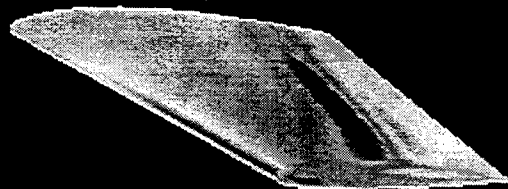
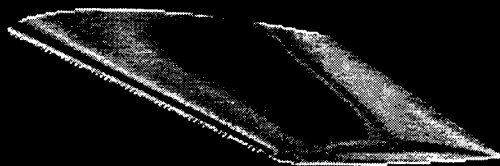
◇ C_p and $(\alpha, C_p, \text{a twist})$ for several geometries

SD for wing via AD, $M=0.84$



◇ C_p and $(d C_p / d \alpha)_{max}$ for several geometries

SD for wing via AD, $M=0.84$



◇ C_p and $(d C_p / d \alpha)_{thick}$ for several geometries

Summary

- ◆ Feasibility of using AD in CFD demonstrated
- ◆ ADIFOR calculated exact geometric SD for grid-flow coupling similar to MDO problem
- ◆ ADIFOR calculated SD through complex algorithm for nonlinear problem
- ◆ ADIFOR processing easier & faster than quasi-analytic method
- ◆ AD competitive with & more accurate than divided differences

Special thanks to...

- ◆ Veer Vatsa for use of TLNS3D code
- ◆ Mary Adams for FAST animation sequences
- ◆ Thomas Roberts for PowerBook movies
- ◆ John Knox for video production
- ◆ Thomas Zang for continued support
- ◆ Laura Hall, Andreas Griewank, and George Corliss for initial training and support with ADIFOR

**Sensitivity Derivatives =
BETTER**

**Multidisciplinary Design
Optimization =
PRODUCTS & PROCESSES**

**Automatic Differentiation =
EASILY, QUICKLY & RELIABLY**

356078

110047

N95-16462

**Automatic Differentiation for Design Sensitivity Analysis of
Structural Systems Using Multiple Processors**

Duc T. Nguyen* , Olaf O. Storaasli† , Jiangning Qin* , and Ramzi Qamar*
Multidisciplinary Design Optimization Branch
Fluid Mechanics and Acoustics Division

P. 32

Automatic differentiation tools (ADIFOR) is incorporated into a finite element based structural analysis program for shape and non-shape design sensitivity analysis of structural systems. The entire analysis and sensitivity procedures are parallelized and vectorized for high performance computation. Small-scale examples to verify the accuracy of the proposed program and a medium-scale example to demonstrate the parallel-vector performance on the multiple Cray-C90- processors are included in the paper.

* Multidisciplinary Parallel-Vector Computation Center, 135 KDH Building, Old Dominion University, Norfolk VA 23529-0241

† Computational Mechanics Branch, NASA Langley Research Center, Hampton, VA 23681

Automatic Differentiation for Design Sensitivity Analysis of Structural Systems Using Multiple Processors

by

Duc T. Nguyen[†], Olaf O. Storaasli[§], Jiangning Qin[†], and Ramzi Qamar[†]

[†] Multidisciplinary Parallel-Vector Computation Center, 135 KDH Building, Old Dominion University, Norfolk, VA 23529-0241

[§] Computational Mechanics Branch, NASA Langley Research Center, Hampton, VA 23681

Abstract

Automatic differentiation tools (ADIFOR) is incorporated into a finite element based structural analysis program for shape and non-shape design sensitivity analysis of structural systems. The entire analysis and sensitivity procedures are parallelized and vectorized for high-performance computation. Small-scale examples to verify the accuracy of the proposed program and a medium-scale example to demonstrate the parallel-vector performance on the multiple Cray-C90 processors are included in the paper.

I. Introduction

Using the familiar finite element procedure^[1], the static equilibrium equations for a structural model can be expressed as

$$[K(b)]_{n \times n} \{z\}_{n \times 1} = \{F\}_{n \times 1} \quad (1)$$

where $[K(b)]$, $\{z\}$ and $\{F\}$ are referred to the stiffness matrix, nodal displacement vector and nodal force vector, respectively. In Eq. (1), "n" represents the active degree-of-freedom of the discretized structural model.

The stiffness matrix $[K(b)]$, in general, is a function of design variable vector $\{b\}$ (where $b \in R^k$). As an example, $\{b\}$ may represent the cross-sectional areas of various truss members, or thickness of plate members (for non-shape type of design variables), or it may also represent the joint coordinates of various nodes of a structure (for shape type of design variables).

A typical constraint, involving a limit on a displacement or a stress component, may be written as

$$g(z, b) \leq 0 \quad (2)$$

For the sake of simplified notation, it is assumed that g depends on only a simple design variable b (i.e. $b \in R^{k=1}$). Using the chain rule of differentiation, one obtains

$$\frac{dg}{db} = \frac{\partial g}{\partial b} + x^T \frac{dz}{db} \quad (3)$$

where x is a vector with components

$$x_i = \frac{\partial g}{\partial z_i} \quad (4)$$

The first term on the right-hand-side of Eq. (3) is usually zero or easy to obtain, thus one discusses only the computation of the second term.

Differentiating Eq. (1) with respect to b , one obtains

$$K * \frac{dz}{db} = \frac{\partial F}{\partial b} - \frac{dK}{db} * z \quad (5)$$

Premultiplying Eq. (5) by $x^T K^{-1}$, one obtains

$$x^T \frac{dz}{db} = x^T K^{-1} \left(\frac{\partial F}{\partial b} - \frac{dK}{db} * z \right) \quad (6)$$

Numerically, the computation of $x^T \frac{dz}{db}$ can be performed in two different ways. The first, called the "direct method", consists of solving Eq. (5) for $\frac{dz}{db}$ and then taking the scalar product with x . The second approach, called the "adjoint method"^[2, 3], defines an adjoint vector λ which is the solution of the system

$$K \lambda = x \quad (7)$$

or

$$\lambda = K^{-1} x \quad (8)$$

or

$$\lambda^T = x^T K^{-1} \quad (\text{since matrix } K \text{ is symmetric}) \quad (9)$$

and thus, Eq. (3) can be re-written as

$$\frac{dg}{db} = \frac{\partial g}{\partial b} + \lambda^T \left(\frac{\partial F}{\partial b} - \frac{dK}{db} * z \right) \quad (10)$$

The solution of Eq. (7) for λ is similar to a solution for displacement under a "dummy" load vector $\{x\}$.

Once, the sensitivity information $\frac{dg}{db}$ has been computed, any gradient based optimization softwares^[4, 5] can be used to obtain a new, improved design.

The focus of this paper is in the parallel computation of $\frac{dz}{db}$ as shown in Eq. (5), and particularly, the computation of the term $\frac{dK}{db}$.

Since in the finite element procedure

$$[K] = \sum_{e=1}^{\# \text{ elements}} [k^{(e)}] \quad (11)$$

Therefore, computation of $\frac{d[K]}{db}$ involves with computation of $\frac{d[k^e]}{db}$ and the latter can be obtained either by

(i) Finite Difference Method

or

(ii) Analytical Method

In the finite difference method, a small perturbation of a design variable is first applied, then approximate derivative (which can be affected by round-off and truncation errors^[3]) can be generated. The analytical method tends to generate very cumbersome expressions for the derivatives. Thus, the objectives of this paper is to use automatic differentiation (ADIFOR) tools^[6] to compute the derivatives of $\frac{d[k^e]}{db}$ in a parallel-vector computer environment.

A brief review of ADIFOR tools^[6] is given in Section 2. Parallel generation and assembly^[7] of the stiffness matrix [K] is presented in Section 3. Parallel-Vector equation solver^[8] which will be used to solve system of Eq. (5) is summarized in Section 4. Numerical examples are presented in Section 5, and conclusions are drawn in Section 6.

II. A Brief Review on Automatic Differentiation^[6]

Automatic Differentiation (AD) is essentially an automatic implementation of the chain rule of differentiation based on tracking the connection between the dependent (or output) and independent (or input) variables.

Typically, to calculate the derivative of any output variable in a computer program with respect to any input variable, one modifies the original program by inserting of specialized instruction which identify the relevant output and input variables.

Automatic differentiation produces exact derivatives, limited only by machine precision. There are two modes of AD. In the forward mode, the chain rule is evaluated from the input to the output. In this mode, the computational cost increases with the number of input variables. In the reverse mode, the chain rule is evaluated from the output to the input.

In order to understand the forward mode in AD, let's refer to Figure 1 where the computation flow to evaluate

$$y_3 = \frac{-20 b_2}{(2 b_1 b_2 + \sqrt{2} b_1^2)} = \frac{-20 b_2}{b_1 (2 b_2 + \sqrt{2} b_1)}$$

is shown in a form of the directed graph.

The derivatives of $\frac{dy_3}{db_2}$ and $\frac{dy_3}{db_1}$ are also shown in a form of the directed graph in Figure 2.

In Figure 2, the connecting link between any 2 vertex represents the chain-rule derivatives. As an example, $\frac{\partial a}{\partial b_2} = 2 b_1$ and $\frac{\partial d}{\partial a} = 1$.

On the other hand, if the reverse mode of differentiation is used to calculate $\frac{dy_3}{db_2}$, then the chain-rule of differentiation will start with the output variable y_3 , and then proceed as following:

$$\begin{aligned} \frac{dy_3}{dd} &= \frac{20 b_2}{(2 b_1 b_2 + \sqrt{2} b_1^2)^2} \\ \frac{dy_3}{da} &= \frac{dy_3}{dd} \frac{\partial d}{\partial a} = \frac{20 b_2 * 1}{(2 b_1 b_2 + \sqrt{2} b_1^2)^2} \\ \frac{dy_3}{dx_2} &= \frac{dy_3}{da} \frac{\partial a}{\partial b_2} + \frac{\partial y_3}{\partial b_2} \\ &= \frac{20 b_2 * 1 * 2 b_1}{(2 b_1 b_2 + \sqrt{2} b_1^2)^2} + \frac{-20}{(2 b_1 b_2 + \sqrt{2} b_1^2)} \end{aligned}$$

It has been concluded from earlier research works^[6, 9, 10] that using automatic differentiation (AD) method, such as ADIFOR tool^[6], will be more computationally efficient than the finite difference method. In most problems, however, analytical method is more efficient than ADIFOR tool (but at the expense of assuming there is no human errors in deriving analytical derivative expressions).

The comparisons of computational costs and the accuracy to evaluate derivative information between the Finite Difference, Analytical and ADIFOR have been discussed^[6, 9, 10]. This paper, therefore, will focus on the issue of incorporating derivative calculation subroutines (generated by ADIFOR) in a parallel-vector high-performance computer environment.

III. Parallel Generation and Assembly on Distributed- and Shared Memory Computers^[7]

The choice of the storage scheme for the global stiffness matrix in any finite element analysis code is based on whether it will save the memory or it will enhance the vector speed, or both. The row-oriented storage scheme^[8] is good for saxpy operation and shared memory type computers, while the skyline storage is good for dot product (daxpy) operation. Moreover, the skyline storage scheme requires less memory and this feature is important for computers with distributed-memory (since each processor usually has less memory capacity as compared to shared-memory computers). Fortunately, the Intel iPSC/860 computers have good vector performance for daxpy operation. In order to use the vector-unrolling technique to improve the vector performance, a block-skyline columns storage and block rows storage schemes for the stiffness matrix is used on the Intel and Cray type computers, respectively (as shown in Figure 3). To simplify the discussion, assuming the global matrix is full and three processors are used to store different portions of the global stiffness matrix.

The size of the block is called k if there are k columns (or k -rows) in each block. It is realized that the choice of k will have the effects on

1. the in-core memory requirement,
2. the vector performance,
3. the communication performance.

For the Intel iPSC/860 parallel computers, the block size in MPFEA is set to be 8. Since each processor only has certain *block-columns* (or block rows) of the global stiffness matrix, the generation and assembly of this matrix can be done in parallel without any communications among processors. The work involved in the generation and assembly procedure can be summarized as (for each processor i , where $i = 1, 2, \dots, NP$):

- Task 1. To identify (but not to search for!) the elements that contribute to the columns (or rows) which belong to processor i .
- Task 2. To generate these elements stiffness matrices.
- Task 3. To assemble the global stiffness matrix with these element stiffness matrices.

It should be noted here that even for the case of nonlinear structural analysis, Task 1 of the above procedure needs to be done only once, while Task 2 and Task 3 have to be performed repeatedly since the global matrix will be updated in each nonlinear iteration.

IV. Parallel-Vector Choleski Method Development⁽⁸⁾

In the sequential Choleski method, a symmetric, positive-definite stiffness matrix, $[K]$, can be decomposed as

$$[K] = [U]^T [U] \quad (12)$$

with the coefficients of the upper-triangular matrix, $[U]$:

$$u_{ij} = 0 \quad \text{for } i > j \quad (13)$$

$$u_{11} = \sqrt{K_{11}}; \quad u_{1j} = \frac{K_{1j}}{u_{11}} \quad \text{for } j \geq 1 \quad (14)$$

$$u_{ii} = \sqrt{K_{ii} - \sum_{k=1}^{i-1} u_{ki}^2} \quad \text{for } i > 1 \quad (15)$$

$$u_{ij} = \frac{K_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj}}{u_{ij}} \quad \text{for } i, j > 1 \quad (16)$$

For example, u_{57} can be computed from Eq. (18) as:

$$u_{57} = \frac{k_{57} - u_{15} u_{17} - u_{25} u_{27} - u_{35} u_{37} - u_{45} u_{47}}{u_{55}} \quad (17)$$

The calculations in Eq. (17) for the term u_{57} (of row 5) only involve columns 5 and 7. Furthermore, the "final value" of u_{57} cannot be computed until the final, updated values of the first four rows have been completed. Assuming that only the first two rows of the factored matrix, $[U]$, have been completed, one still can compute the second partially-updated value of u_{57} as designated by superscript (2):

$$u_{57}^{(2)} = k_{57} - u_{15} u_{17} - u_{25} u_{27} \quad (18)$$

If row 3 has also been completely updated, then the third partially-updated value of u_{57} can be calculated as:

$$u_{57}^{(3)} = u_{57}^{(2)} - u_{35} u_{37} \quad (19)$$

This observation suggests an efficient way to perform Choleski factorization in parallel on NP

processors. For example, each row of the coefficient stiffness matrix, $[K]$, is assigned to a separate processor.

From Eq. (17), assuming $NP = 4$, it is seen that row 5 cannot be completely updated until row 4 has been completely updated. In general, in order to update the i^{th} row, the previous $(i-1)$ rows must already have been updated. For the above reasons, any NP consecutive rows of the coefficient stiffness matrix, $[K]$, will be processed by NP separate processors. As a consequence, while row 5 is being processed by a particular processor, say processor 1, then the first $(5-NP)$ rows have already been completely updated. Thus, if the i^{th} row is being processed by the p^{th} processor, there is no need to check every row (from row 1 to row $i-1$) to make sure they have been completed. It is safe to assume that the first $(i-NP)$ rows have already been completed as shown in the triangular cross-hatched region of Figure 4.

Synchronization checks are required only for the rows between $(i-NP + 1)$ and $(i-1)$ as shown in the rectangular solid region of Figure 4. Since the first $(i-NP)$ rows have already been completely factored, the i^{th} row can be "partially" processed by the p^{th} processor as shown in Eq. (18, 19).

V. Numerical Applications

Different finite element types (such as 2-D Truss, and Plate/Shell elements) and different type of design variables (such as cross-sectional areas, joint coordinates of truss elements and thickness of plate elements) are considered in this section. The first two examples are small-size for the purpose of verifying the accuracy of derivatives $(d [k^{(e)}] / d b)$ generated by ADIFOR^[6] as compared to the ones obtained by finite difference technique. The last example is medium-size for the purpose of evaluating the parallel-vector performance of the entire finite element and Design Sensitivity Analysis (DSA) process.

Example 1: Plate-Structure With (Non-Shape) Thickness Design Variable

In this example, 32 plate elements^[11] are used, a point force is applied at the center of the fixed plate (see Figure 5). Thickness of a plate is selected as (non-shape) design variable in this case. The original thickness is 0.03 and a perturbation of 0.5% is used in the finite (central) difference scheme.

The derivatives of element stiffness matrix (in global reference and using ADIFOR) with respect to the thickness t for typical members such as members 5, 12, and 19 are presented in Table 1. These derivatives are in good agreement with the ones obtained by finite (central) difference scheme.

Example 2: Truss-Structure With (Shape) Joint Coordinate Design Variables

In this example, a 1 bay x 1 story truss structure is shown in Figure 6. This small-scale structure has 4 joints and 5 members. All joint x -coordinates of this structure are selected as (shape) design variables. A horizontal force F is applied at node 1. The dimensions for each base and height of this structure are 12" and 9", respectively. Young modulus and cross-sectional area are 29000 Ksi and 4 in², respectively. A perturbation of 1% is used in the finite (central) difference scheme. The derivatives of element stiffness matrix (in global reference and using ADIFOR) with respect to a typical x -coordinate of joint 2 for members 1 and 5 are presented in Table 2. Again, these derivatives are in good agreements with the ones obtained by finite (central)

difference scheme.

Example 3: A 2-D Truss Structure With 80 Bays and 190 Stories

In this example, a 80 bay x 190 story truss structure is also shown in Figure 6. A horizontal force F is applied at node 100. All other data are the same as in Example 2. There are 96 cross-sectional areas selected as (non-shape) design variables in this example. This structure has 60,990 elements. The resulted structural stiffness matrix has 30,780 degree-of-freedom. Using the variable bandwidth storage scheme^[8] will require a real 1-dimensional array with 5,171,574 words to store the stiffness matrix in the core memory. The average bandwidth for this stiffness matrix is 168.

The performance of the entire finite element analysis and design sensitivity analysis (using ADIFOR tool) on 1, 8, and 16 Cray-C90 processors are shown in Table 3. The total speed-up for the ENTIRE PROCESS are 7.32 and 12.93 when 8 and 16 Cray-C90 processors are used, respectively.

VI. Conclusions

Based upon the numerical results presented in this paper, the following conclusions can be made:

1. Automatic Differentiation (ADIFOR)^[6] tool has been successfully applied to both simple (TRUSS) and complex PLATE/SHELL^[11] finite elements.
2. Both *non-shape* and *shape* design variables can be successfully treated.
3. For the first time (to the authors' knowledge), ADIFOR tool can be applied in a parallel-vector computer environment for *non-shape* and *shape* sensitivity analysis.
4. The *entire* finite element and sensitivity analysis can be done with excellent parallel and vector speed (using all 16 Cray-C90 processors).

VII. Acknowledgments

The financial support from NASA grant NAG1-858 are acknowledged. The authors are also deeply indebted to Drs. L. Green, P. Newman, J. Barthelemy (all from NASA Langley Research Center), C. Bischof (from Argonne National Laboratory) and A. Carle (from Rice University) for helpful discussions during the ADIFOR user workshop (September 13-14, 1993), held at Building 1192C-E, the CFD Laboratory, NASA LaRC). Helpful discussions with Dr. A. Tessler on using his plate/shell element (NASA Langley Research Center) is also appreciated.

VIII. References

1. T.J.R. Hughes, *The Finite Element-Method*, Prentice-Hall, Inc., (1987).
2. J.S. Arora and E.J. Haug, *Applied Optimal Design*, John Wiley & Sons, Inc., (1979).
3. R.T. Haftka, Z. Gürdal, and M.P. Kamat, *Elements of Structural Optimization*, Kluwer

Academic Publishers (1990).

4. R. Thareja and R.T. Haftka, "A Modified Version of NEWSUMT For Inequality and Equality Constraints," VPI Report 148, (March 1985).
5. G.N. Vanderplaats, "CONMIN: A Fortran Program for Constrained Function Minimization", NASA-TM X-62282, (1973).
6. C.H. Bischof and A. Griewank, "ADIFOR: A Fortran System For Portable Automatic Differentiation", Proceedings the 4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, pp. 433-441, AIAA 92-4744-CP, (September 1992).
7. J. Qin and D. T. Nguyen, "A New Parallel-Vector Finite Element Analysis Software on Distributed Memory Computers," Proceedings of the AIAA/ASME/ASCE/AHS 34th SDM Conference, La Jolla, CA (April 19-22, 1993).
8. T.K. Agarwal, O.O. Storaasli, and D.T. Nguyen, "A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers," Proceedings of the AIAA/ASME/ASCE/AHS 31th SDM Conference, Long Beach, CA (April 2-4, 1990).
9. J.F. Barthelemy and L.E. Hall, "Automatic Differentiation As A Tool In Engineering Design," NASA-TM 107661, (August, 1992).
10. C. Bischof, G. Corliss, L. Green, A. Griewank, K. Haigler and P. Newman, "Automatic Differentiation of Advanced CFD Codes for Multidisciplinary Design." Computing Systems in Engineering, Vol. 3, No. 6, pp. 625-637, (1992).
11. A. Tessler, "A C⁰ Anisoparametric Three-Node Shallow Shell Element for General Shell Analysis," MTL-TR-89-72, (August 1989).

Figure 1: Computational Graph for $y_3 = \frac{-20 b_2}{(2 b_1 b_2 + \sqrt{2} b_1^2)} = \frac{-20 b_2}{b_1 (2 b_2 + \sqrt{2} b_1)}$

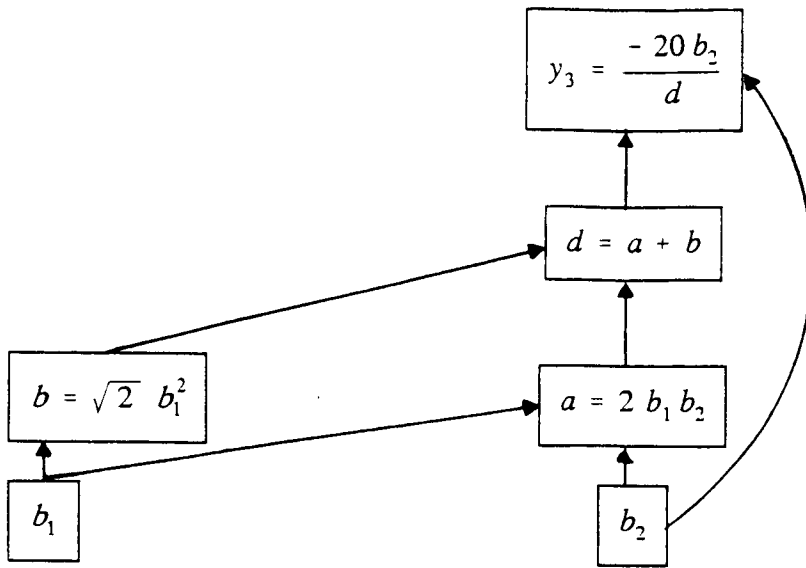


Figure 2: Computational Graph for $\frac{dy_3}{db_2} = \frac{40 b_2 b_1}{(2 b_1 b_2 + \sqrt{2} b_1^2)^2} + \frac{-20}{(2 b_1 b_2 + \sqrt{2} b_1^2)}$

$$\frac{dy_3}{db_2} = \left(\frac{20 b_2}{d^2} \right) (1) (2 b_1) + \left(-\frac{20}{d} \right)$$

$$\frac{dy_3}{db_1} = \frac{(20 * b_2) (2 b_2 + 2 \sqrt{2} b_1)}{[b_1 (2 b_2 + \sqrt{2} b_1)]^2}$$

$$\frac{dy_3}{db_1} = \left(\frac{20 b_2}{d^2} \right) (1) (2 b_2) + \left(\frac{20 b_2}{d^2} \right) (1) (2 \sqrt{2} b_1)$$

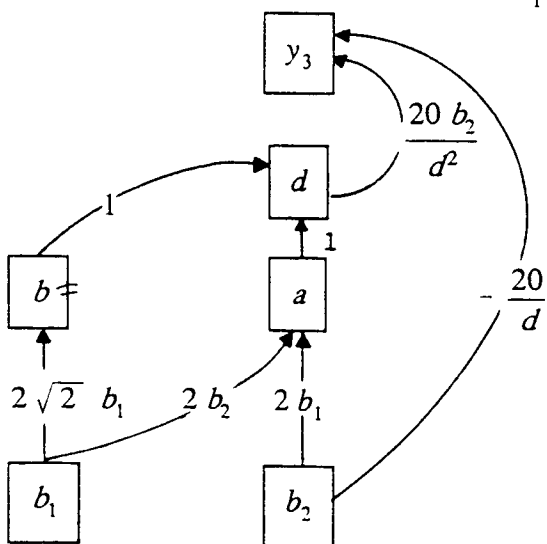


Figure 3. Block-skyline columns storage and block rows storage schemes

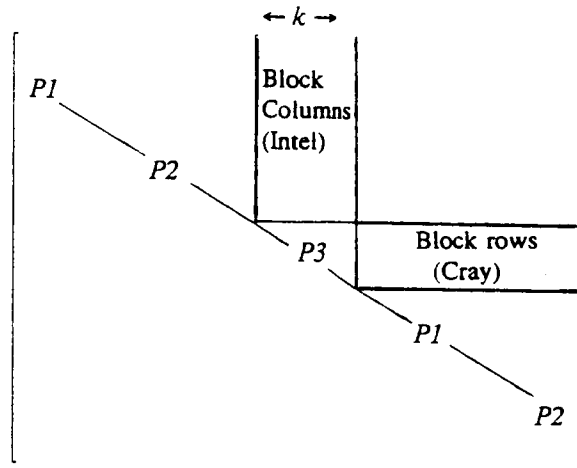


Figure 4: Information required to update row i

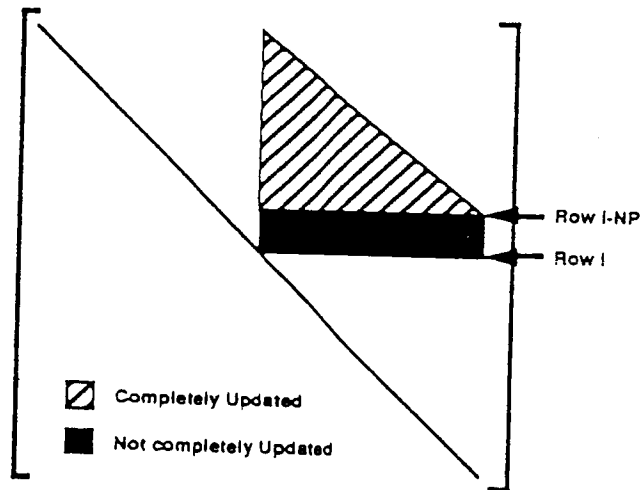


Figure 5: Clamped Plate - Structure

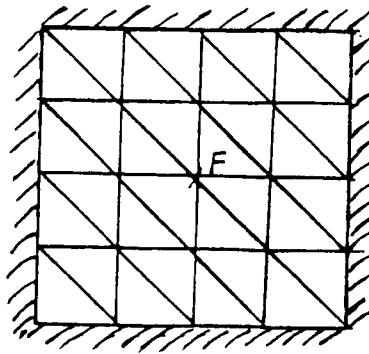
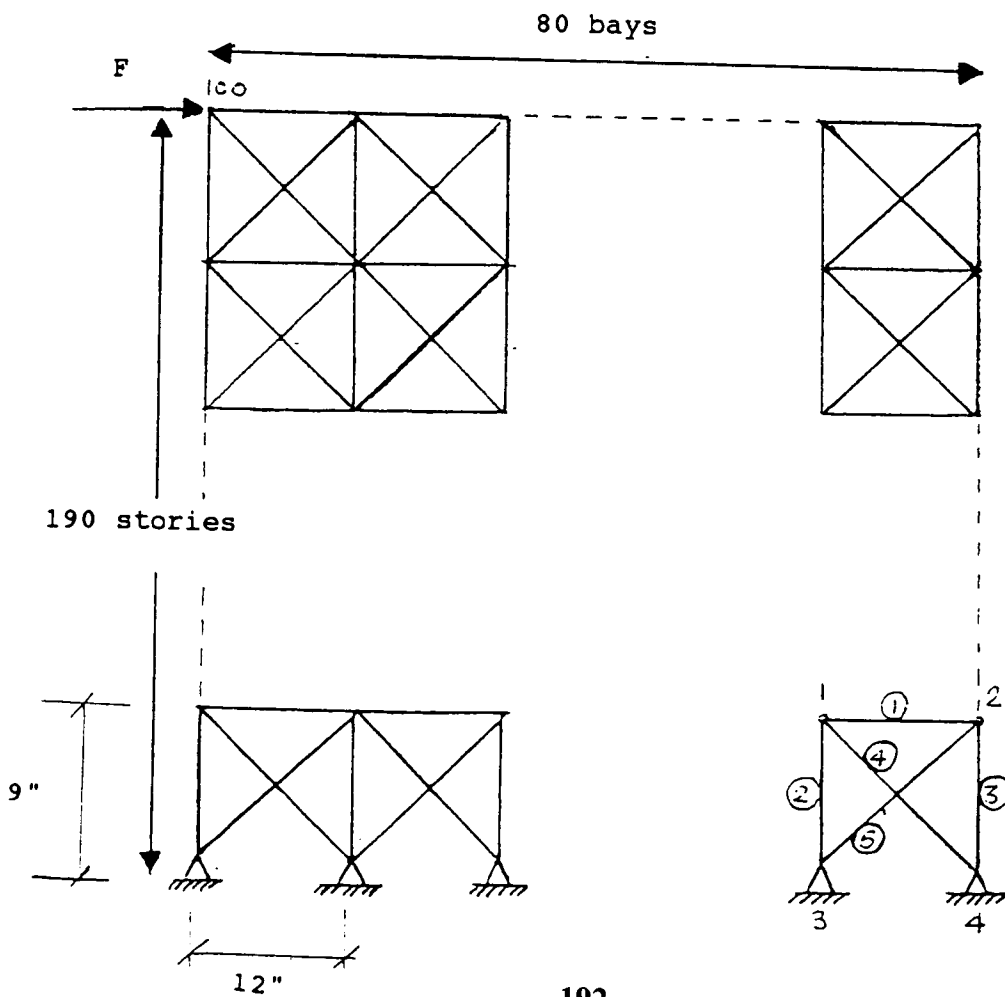


Figure 6: 2-D Truss Structure



$$\frac{\partial [K^{(5)}]}{\partial t} = [5576.925 , - 4780.2198 , 0 , 0 , 0 , - 15934.066 , \dots]$$

$$\frac{\partial [K^{(12)}]}{\partial t} = [15934.068 , 5576.923 , 0 , 0 , 0 , - 5576.923 , \dots]$$

$$\frac{\partial [K^{(19)}]}{\partial t} = [21510.99 , 5576.925 , 0 , 0 , 0 , 8.268E-12 , \dots]$$

Table 1: ADIFOR Derivatives of Plate Element Stiffness Matrix with Respect to Thickness (Non-shape) Design Variable

Table 2: ADIFOR Derivatives of Truss Element Stiffness Matrix with Respect to x-coordinate of Joint 2 (Shape) Design Variable.

el. stiff [k] for member 1

0.966667E+04	0.000000E+00	-0.966667E+04	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
-0.966667E+04	0.000000E+00	0.966667E+04	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Gradient of stiff [k] ^{w.r.t} DV 2 = $\frac{\partial [k^{(1)}]}{\partial x_2}$

-805.555555555556	0.	805.555555555556	0.
0.	0.	0.	0.
805.555555555556	0.	-805.555555555556	0.
0.	0.	0.	0.

el stiff [k] for member 5

0.494933E+04	0.371200E+04	-0.494933E+04	-0.371200E+04
0.371200E+04	0.278400E+04	-0.371200E+04	-0.278400E+04
-0.494933E+04	-0.371200E+04	0.494933E+04	0.371200E+04
-0.371200E+04	-0.278400E+04	0.371200E+04	0.278400E+04

Gradient of stiff [k] w.r.t DV 2 = $\frac{\partial [k^{(5)}]}{\partial x_2}$

32.9955555555555	-284.586666666667	-32.9955555555555	284.586666666667
-284.586666666667	-445.440000000000	284.586666666667	445.440000000000
-32.9955555555555	284.586666666667	32.9955555555555	-284.586666666667
284.586666666667	445.440000000000	-284.586666666667	-445.440000000000

Table 3: Parallel-Vector Performance For DSA of 80 Bays x 190 Stories Truss Structure Using ADIFOR Tool on Multiple Cray-C90 Processors

Tasks	Number of Cray-C90 Processors			Speed-Up Factors	
	1 proc.	8 proc.	16 proc.	8 proc.	16 proc.
(A)	0.4855 ^{sec}	0.09954 ^{sec}	0.07854 ^{sec}	4.88	6.18
(B)	0.9582 ^{sec} (0.9906*)	0.1320 ^{sec} (0.1433*)	0.0731 ^{sec} (0.1547*)	7.26	13.11
(C)	2.6290 ^{sec}	0.3568 ^{sec}	0.2026 ^{sec}	7.37	12.98
(D)	0.1019 ^{sec}	0.1015 ^{sec}	0.1018 ^{sec}	N/A	N/A
(E)	2.3717 ^{sec}	0.3034 ^{sec}	0.1558 ^{sec}	7.82	15.22
(F)	9.6934 ^{sec}	1.2128 ^{sec}	0.6047 ^{sec}	7.99	16.03
Entire Process	16.2740 ^{sec}	2.2221 ^{sec}	1.2591 ^{sec}	7.32	12.93

Notes:

- (A) To generate column heights of stiffness matrix
- (B) To generate and assemble stiffness matrix
- (C) To factorize stiffness matrix
- (D) To get static (forward/backward) solution (sequential computation)
- (E) To generate the right-hand-side vectors for sensitivity equations
- (F) To solve for displacement sensitivity vectors

* Wall-Clock-Time

Automatic Differentiation for Design Sensitivity Analysis of Structural Systems Using Multiple Processors

by

Duc T. Nguyen[†], Olaf Storaasli[§],
Jiangning Qin[†], and Ramzi Qamar[†]

[†]Multidisciplinary Parallel-Vector Computation Center,
135 KDH Building,
Old Dominion University,
Norfolk, VA 23529-0241

[§]Computational Structures Branch,
NASA Langley Research Center,
Hampton, VA 23681

OBJECTIVES

1. To obtain accurate derivatives of complex finite elements and/or complex design variables
2. Design variables can be either non-shape (such as areas, thickness) or shape types (such as joint coordinates)
3. The entire solution process should be parallelized and vectorized to reduce solution time
4. Numerical validation and performance evaluation for the proposed procedure.

MOTIVATION

1. Analytical (hand-coded) derivatives are not feasible for complex finite elements and shape variables
2. Finite difference derivatives are expensive and can be inaccurate

APPROACH USED

1. Utilizing Automatic Differentiation (ADIFOR) tool

*developed by ANL & CRPC at Rice
Application of Back from AAOB & USB at LaRC*

2. Parallelizing and Vectorizing every step of the entire solution process

UNIQUE FEATURES OF THIS WORK

1. Both simple and complex finite elements (2-D truss, and 3-D plate/shell) are treated.
2. Both (non-shape) design variables (such as areas of truss members, or thickness of plate and shell members), and shape design variables (such as joint coordinates) are considered
3. The entire solution process has been parallized and vectorized
4. EXCELLENT speed-up has been achieved even on "small-scale" example.

GENERAL FORMULATION FOR DESIGN SENSITIVITY ANALYSIS (D.S.A.)

- Equilibrium Equations

$$[K] \{Z\} = \{F\} \quad (1)$$

Take derivatives of both sides of Equation (1) with respect to design variable vector b

$$\frac{\partial [K]}{\partial b} * \{Z\} + [K] * \frac{\partial \{Z\}}{\partial b} = [0] \quad (2)$$

- Sensitivity Equations

$$[K] \frac{\partial \{Z\}}{\partial b} = - \frac{\partial [K]}{\partial b} * \{Z\} \quad (3)$$

using "NEW" parallel Gen. + Assem. technique!

Note:

Parallel-Vector Eq. Solver is needed for CSM & CFD !

D.S.A. FOR 2-D TRUSS ELEMENT

$$\begin{aligned}
 [k]_{Global} &= \frac{EA}{L} \\
 &= \frac{EA}{L} * [T]_{4 \times 4} \\
 &= \frac{EA}{L} \begin{bmatrix} C_x^2 & C_x C_y & -C_x^2 & -C_x C_y \\ C_x C_y & C_y^2 & C_x C_y & -C_y^2 \\ -C_x^2 & C_x C_y & C_x^2 & C_x C_y \\ -C_x C_y & -C_y^2 & C_x C_y & C_y^2 \end{bmatrix}_{4 \times 4}
 \end{aligned}$$

where

$$\left\{ \begin{aligned}
 L &= \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2} \\
 C_x &= \frac{x_k - x_j}{L} \\
 C_y &= \frac{y_k - y_j}{L}
 \end{aligned} \right.$$

(A) Non-Shape Design Variables

$$\frac{\partial [k^{(e)}]}{\partial \vec{b}} = \frac{\partial [k^{(e)}]_{Global}}{\partial \vec{A}} = \frac{E}{L} * [T] = \text{EASY !}$$

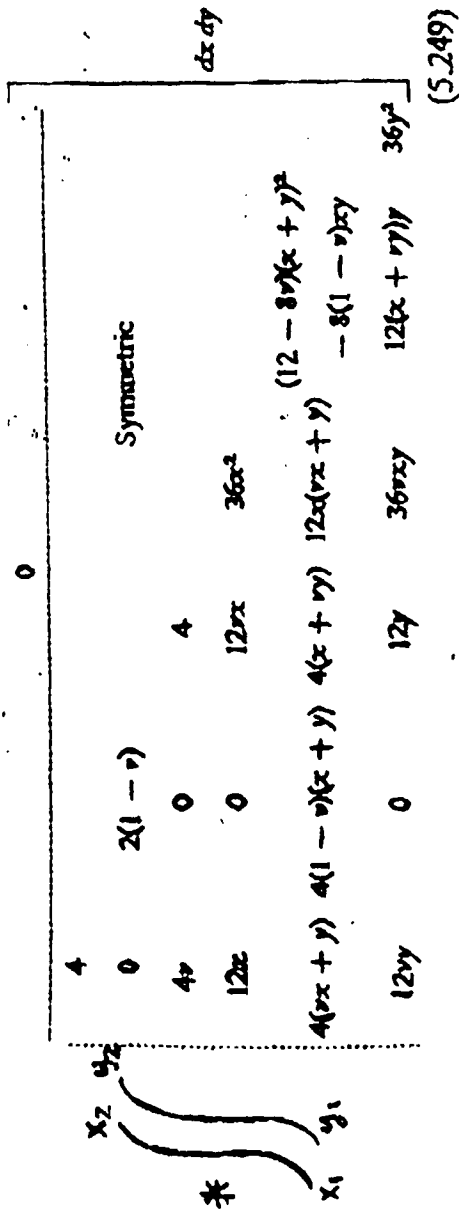
(B) Shape Design Variables

$$\frac{\partial [k^{(l)}]_{Global}}{\partial \left\{ \begin{array}{c} x_j \\ x_k \\ y_j \\ y_k \end{array} \right\}} = \text{VERY TEDIOUS !}$$

D.S.A. FOR TRIANGULAR PLATE BENDING ELEMENT

(Reference: Preziemienicki's Book)

$$[k^{(e)}]_{Local} = \int_0^1 D^T \kappa D dV = \frac{Et^3}{12(1-\nu^2)}$$



	4	0	2(1-ν)		Symmetric	
x_2	0	4ν	0	4		
y_2	12x	0	0	12x	36x ²	
*	$4(x+y)$	$4(1-\nu)(x+y)$	$4(x+y)$	$4(x+y)$	$12x(\nu x+y)$	$(12-8\nu)(x+y)^2$
x_1	12xy	0	0	12y	36xy	$-8(1-\nu)xy$
y_1					$12(x+y)y$	$36y^2$

(5.249)

Then

$$[k^{(e)}]_{Global} = [k^{(e)}]_{Local} * [T]$$

Functions of element nodal coordinates also!

(A) Non-Shape Design Variables

$$\frac{\partial [k^{(e)}]_{Global}}{\partial t} = E A S Y !$$

(B) Shape Design Variables

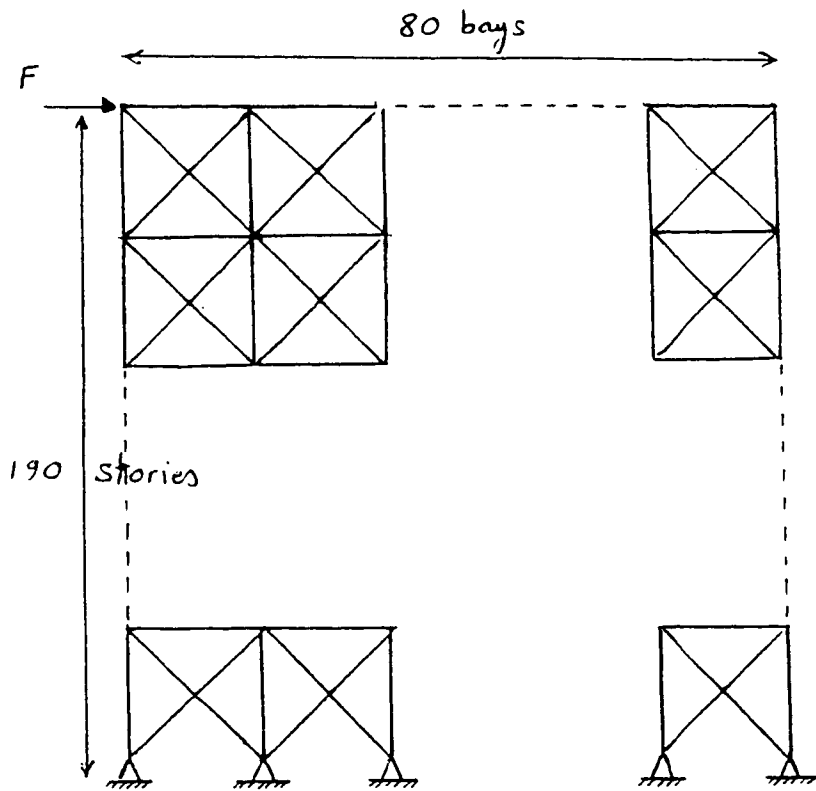
$$\frac{\partial [k^{(e)}]_{Global}}{\partial \left\{ \begin{array}{c} x_i \\ x_j \\ x_k \\ \cdot \\ \cdot \\ \cdot \\ z_i \\ z_j \\ z_k \end{array} \right\}} = V E R Y T E D I O U S !$$

D.S.A. FOR TRIANGULAR PLATE/SHELL BENDING ELEMENT

(Reference: Alex Tessler's Element)

Note: In this work, Alex Tessler's Plate/Shell finite element is used.

APPLICATIONS



Example 1 (80 bays, 190 stories)

NEL = 60,990 elements (TRUSS)

NEQ = 30,780 equations

NTERMS = 5,171,574 terms

AVEBW = 168 average bandwidth

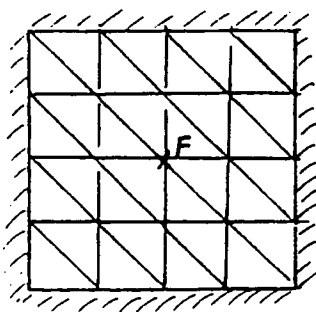
NDV = 96 Non-shape
design variables

Example 2 (1 bay, 1 story)

NEL = 5 elements (TRUSS)

NEQ = 4

NDV = 8 shape design variables



Example 3 (Plate structure)

NEL = 32 elements (Alex Tosler's
Plate/Shell)

NDV = 75 shape
design variables

NUMERICAL RESULTS AND PARALLEL PERFORMANCE

Tasks	Number of Cray - C90 Processors			Speed-Up Factors	
	1 proc.	8 proc.	16 proc.	8 proc.	16 proc.
(A)	0.4855 ^{sec}	0.09954 ^{sec}	0.07854 ^{sec}	4.88	6.18
(B)	0.9582 ^{sec} (0.9906*)	0.1320 ^{sec} (0.1433*)	0.0731 ^{sec} (0.1547*)	7.26	13.11
(C)	2.6290 ^{sec}	0.3568 ^{sec}	0.2026 ^{sec}	7.37	12.98
(D)	0.1019 ^{sec}	0.1015 ^{sec}	0.1018 ^{sec}	N/A	N/A
(E)	2.3717 ^{sec}	0.3034 ^{sec}	0.1558 ^{sec}	7.82	15.22
(F)	9.6934 ^{sec}	1.2128 ^{sec}	0.6047 ^{sec}	7.99	16.03
Entire Process	16.2740 ^{sec}	2.2221 ^{sec}	1.2591 ^{sec}	7.32	12.93

```

nel,ndofpe,nodes,ndofpn,nunrol,nummat,ireal,nbays,      nstory,ndv
7, 4, 2*2, 8, 1000, 0, 10, 300, (1000)
design variable, total memory needed= 1000, 7881649
max. wall clock time for gen+assem = 0.250323822
(z)/d(b) with respect to DV # 1000
d(z)/d(b) = 0.463915E-02 0.197448E-03 0.463915E-02 0.127947E-03 0.463915E-02
d(z)/d(b) = 0.584470E-04 0.463915E-02-0.110535E-04 0.463915E-02-0.805540E-04
d(z)/d(b) = 0.463915E-02-0.150054E-03 0.463915E-02-0.219555E-03 0.463915E-02
d(z)/d(b) = -0.289055E-03 0.463915E-02-0.358556E-03 0.463915E-02-0.428056E-03
ME, time for generate SD=1, 2.465475E-2
ME, time for generate K =1, 0.250328442
ME, time for Factori. =1, 0.234019218
ME, time for Solution =1, 2.2428906000002E-2
ME, time for (dK/db)*X =1, 6.526009938
ME, time for dX/db =1, 21.726178002
** Time in boundc =4.737786E-3
** Time in jointc =3.029639999999999E-4
** Time in apload =4.830000000000001E-5
** Time in elconn =3.33129E-3
** Time in materp =4.7050644E-2
** Time in colht =0.152899086
TOTAL TIME:(nel,neq,ielm,nterms)28.991996244, 12300, 6600, 12300, 186532

```

```

nel,ndofpe,nodes,ndofpn,nunrol,nummat,ireal,nbays,      nstory,ndv
, 4, 2*2, 8, 1000, 0, 10, 300, (1000)
design variable, total memory needed= 1000, 7881649
max. wall clock time for gen+assem = 0.143736066
d(z)/d(b) with respect to DV # 1000
d(z)/d(b) = 0.463915E-02 0.197448E-03 0.463915E-02 0.127947E-03 0.463915E-02
d(z)/d(b) = 0.584470E-04 0.463915E-02-0.110535E-04 0.463915E-02-0.805540E-04
d(z)/d(b) = 0.463915E-02-0.150054E-03 0.463915E-02-0.219555E-03 0.463915E-02
d(z)/d(b) = -0.289055E-03 0.463915E-02-0.358556E-03 0.463915E-02-0.428056E-03
ME, time for generate SD=1, 2.466528E-2
ME, time for generate K =1, 0.125706372
ME, time for Factori. =1, 0.141787986
ME, time for Solution =1, 2.2426734E-2
ME, time for (dK/db)*X =1, 3.275844468
ME, time for dX/db =1, 10.83518028
ME, time for generate SD=2, 1.8168000000429E-5
ME, time for generate K =2, 0.12609603
ME, time for Factori. =2, 0.141645408
ME, time for Solution =2, 5.1287999999872E-5
ME, time for (dK/db)*X =2, 3.28376691
ME, time for dX/db =2, 10.836083088
** Time in boundc =4.723914E-3
** Time in jointc =3.02904000000002E-4
** Time in apload =4.83480000000004E-5
** Time in elconn =3.331998E-3
** Time in materp =4.7076264E-2
* Time in colht =3.3666912000001E-2
TOTAL TIME:(nel,neq,ielm,nterms)14.514775074, 12300, 6600, 6150, 186532
TOTAL TIME:(nel,neq,ielm,nterms)14.528120334, 12300, 6600, 6181, 186532

```

```

el,ndofpe,nodes,ndofpn,nunrol,nummat,ireal,nbays,      nstory,ndv
7, 4, 2*2, 8, 1000, 0, 10, 300, 1000
#design variable, total memory needed= 1000, 7881649
max. wall clock time for gen+assem = 0.106087758
d(z)/d(b) with respect to DV # 1000
d(z)/d(b) = 0.463915E-02 0.197448E-03 0.463915E-02 0.127947E-03 0.463915E-02
d(z)/d(b) = 0.584470E-04 0.463915E-02-0.110535E-04 0.463915E-02-0.805540E-04
d(z)/d(b) = 0.463915E-02-0.150054E-03 0.463915E-02-0.219555E-03 0.463915E-02
d(z)/d(b) = -0.289055E-03 0.463915E-02-0.358556E-03 0.463915E-02-0.428056E-03
ME, time for generate SD=3, 2.441805E-2
ME, time for generate K =3, 8.4435756E-2
ME, time for Factori. =3, 9.39547440000001E-2
ME, time for generate SD=1, 2.20686000000003E-4
ME, time for generate SD=2, 2.52960000000008E-5
ME, time for generate K =1, 8.41718460000001E-2
ME, time for generate K =2, 8.452527E-2
ME, time for Factori. =1, 9.3851616E-2
ME, time for Factori. =2, 9.39392700000001E-2
ME, time for Solution =1, 6.43739999999923E-5
ME, time for Solution =2, 5.4413999999917E-5
ME, time for (dK/db)*X =1, 2.156280738
ME, time for (dK/db)*X =2, 2.168834262
ME, time for dX/db =1, 7.22125122
E, time for dX/db =2, 7.196945016
E, time for Solution =3, 2.2304856E-2
E, time for (dK/db)*X =3, 2.163674568
ME, time for dX/db =3, 7.211424354
** Time in boundc =4.723506E-3
** Time in jointc =3.02075999999998E-4
** Time in apload =4.83420000000007E-5
** Time in elconn =3.328746E-3
** Time in materp =4.6941834E-2
** Time in colht =1.0283598E-2
TOTAL TIME:(nel,neq,ielm,nterms)9.621492162, 12300, 6600, 4100, 186532
TOTAL TIME:(nel,neq,ielm,nterms)9.607627608, 12300, 6600, 4131, 186532
TOTAL TIME:(nel,neq,ielm,nterms)9.721047816, 12300, 6600, 4131, 186532

```

CONCLUSIONS

1. Automatic Differentiation (ADIFOR) tool has been successfully applied to both simple (TRUSS) and complex (Alex Tessler's PLATE/SHELL) finite elements
2. Both non-shape and shape design variables can be successfully treated.
3. For the first time (to the author's knowledge), ADIFOR tool can be applied in a parallel-vector computer environment for non-shape and shape sensitivity analysis.
4. The entire finite element + sensitivity analysis can be done with excellent parallel and vector speed (using all 16 Cray-C90 processors)

SESSION 6 Mosaic and the World Wide Web

Chaired by

Clyde R. Gumbert and John W. McManus

- 6.1 Introduction to the World Wide Web and Mosaic -Jim Youngblood
- 6.2 Use of World Wide Web and NCSA Mosaic at Langley -Michael Nelson
- 6.3 How To Use the WWW To Distribute Scientific & Technical Information (STI)
-Donna Roper

Introduction to the World Wide Web and Mosaic

by Jim Youngblood, Lockheed Langley Program Office

Special thanks to Earl Spratley of Lockheed Langley Program Office for assistance with the graphics.

6/13/94 5:30 p. m.

IMPORTANT: This document is a hypertext file. If you are reading it in printed form you can get an electronic version by using your Mosaic browser's "Open URL" feature. This document's URL is "<http://sti.larc.nasa.gov/demos/mosaic-general.html>". The electronic version contains Hyperlinks that allow you to access reference documents in other parts of the World Wide Web. (All of this is explained in more detail below.)

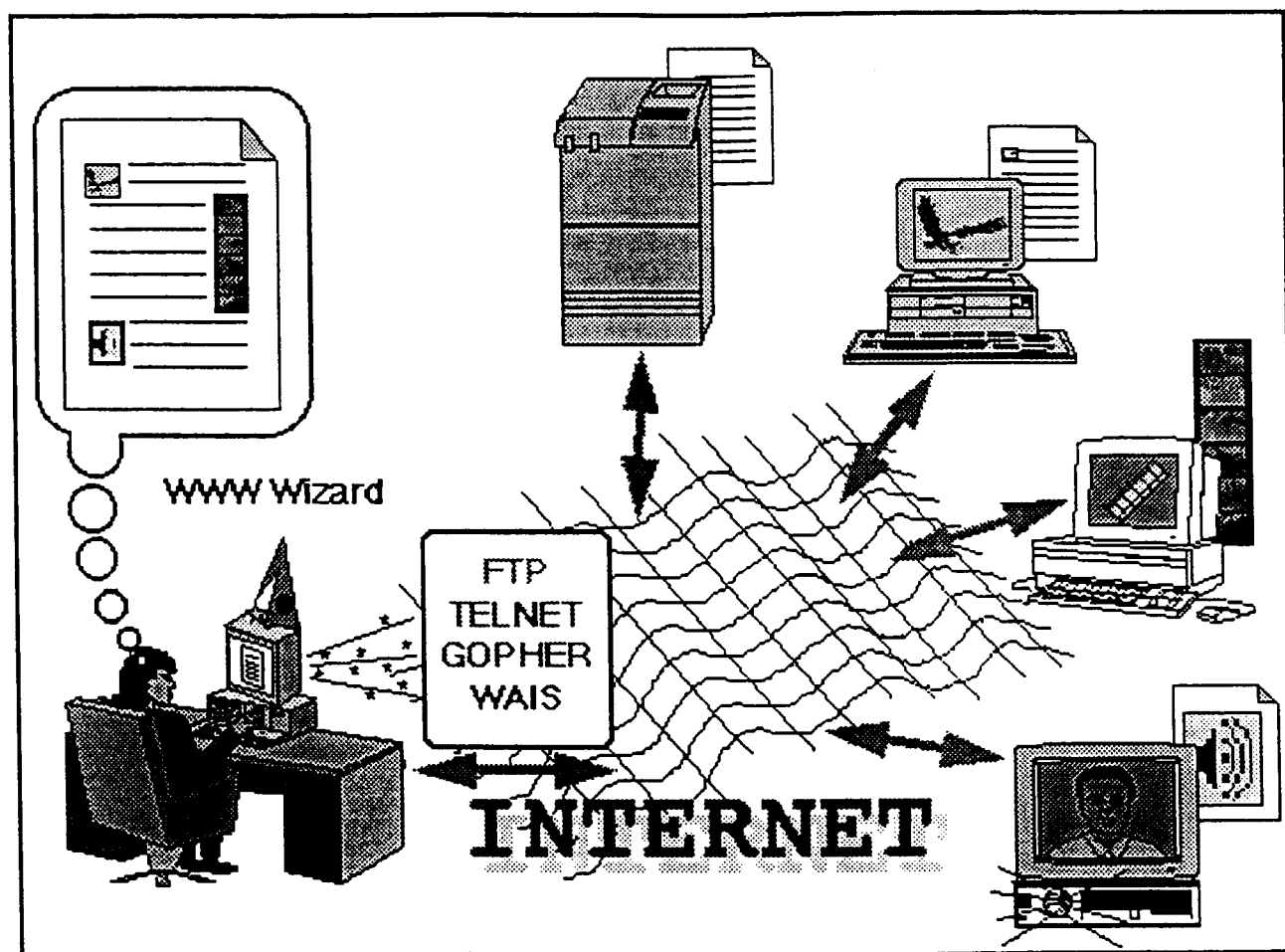
Introduction

This tutorial provides an introduction to some of the terminology related to the use of the World Wide Web and Mosaic. It is assumed that the user has some prior computer experience. References are included to other sources of additional information.

The concepts are:

- The World Wide Web
- Browsers
- Mosaic
- Hypertext
- Hypermedia
- Distributed Hypermedia
- Hyperlinks
- HTML
- URL
- Hotlist
- Hints

If you are reading this document from within Mosaic and you are familiar with some of these concepts and want to skip to an unfamiliar section just place your mouse cursor on the section you wish to read and click the mouse button. If you are reading this document in printed form, the sections proceed in the order given above.



What is the World Wide Web (WWW or W3)?

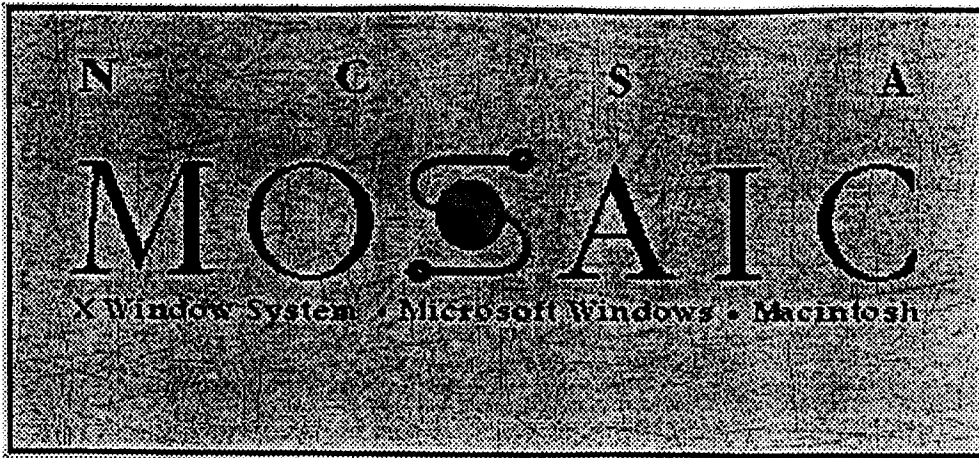
The world wide web was first conceived at the CERN high energy physics research laboratory in Switzerland as a way to quickly share physics research results over the Internet. The shared data was often graphical in nature so existing methods of distributing text were not adequate. CERN defined standards for uniform access methods to all forms of media on the net. There are several different WWW clients; Mosaic is emerging as the most popular. The WWW attempts to find uniform ways to access all of the current Internet resources including:

- Gopher (An on-line card catalog of many on-line libraries.)
 - WAIS (An on-line catalog browser and retrieval mechanism)
 - FTP (File Transfer Protocol) -- A way to transfer files to and from other computers to your computers.)
 - Usenet (The worlds LARGEST computer bulletin board)
 - telnet (A way to log into other computers)
 - hytelnet (A menu driven version of telnet)
 - hyper-g (A hypermedia system built on existing large databases, Computer Aided Instruction lessons and a general purpose hypermedia encyclopedia)
 - techinfo (Another Internet based information -- similar to Gopher)
 - texinfo (Based on Donald Knuth's TeX typesetting system, texinfo allows one file to produce both on-line help files and a printed manual)
 - man pages --UNIX manual pages on-line (help files)
 - hypertext documents
 - "Phone book" services (On-line "White" and "Yellow" pages)
-

Browsers

A browser is simply a software application that recognizes the standards that define the World Wide Web. Mosaic is **not** the only browser for the World Wide Web. Some of the other browsers are:

- Cello for Microsoft Windows
 - DosLynx for MSDOS
 - Samba and MacWeb for the Macintosh
 - Chimera, tkWWW and MidasWWW for X Windows System
 - Lynx text mode browser for UNIX
-

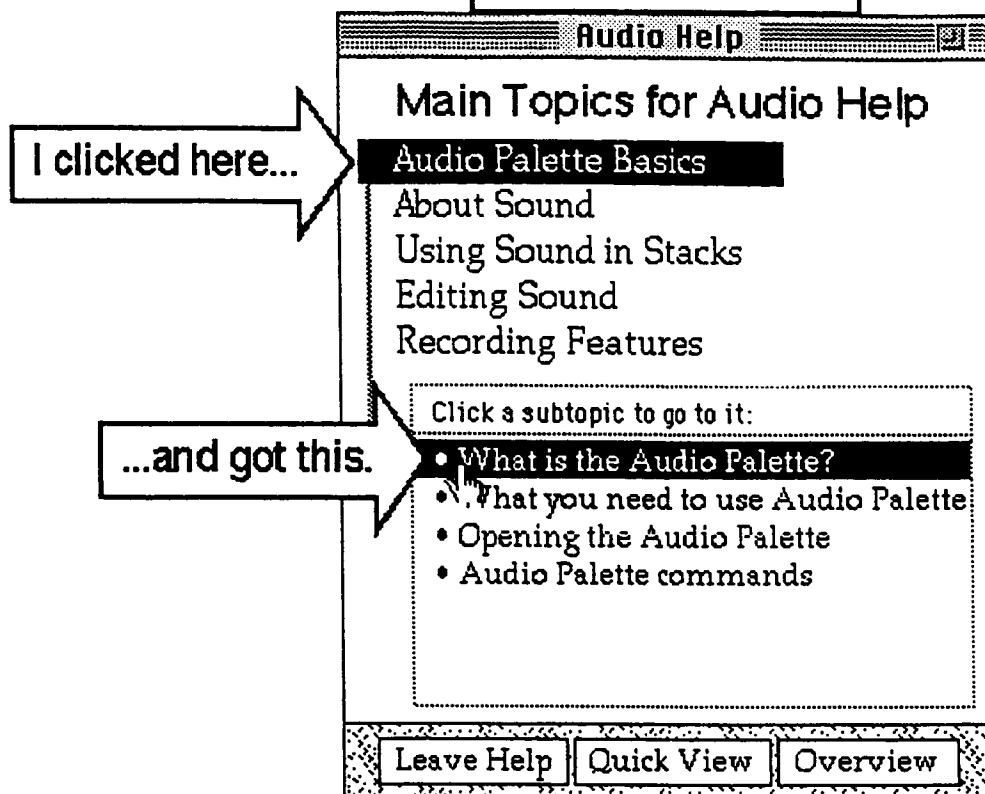


What is Mosaic?

Mosaic is a distributed hypermedia browser for the World Wide Web (WWW or W3). Mosaic was originally developed in the USA at the National Center for Supercomputer Application (NCSA) at the University of Illinois at Urbana/Champaign, and is in the public domain. Mosaic was originally X-mosaic for X Window System for UNIX. Mosaic has become so popular that it has been renamed from X-mosaic because it is now available for X Window System, PCs and Macs. Mosaic is available in version 2.0 for X Window System and PCs. Version 2.0 Alpha for the Mac was released on June 10, 1994. It is not known how stable and usable this release is. Version 1.0.3 for the Mac is the current fully released version. This version does not have "Forms Support".

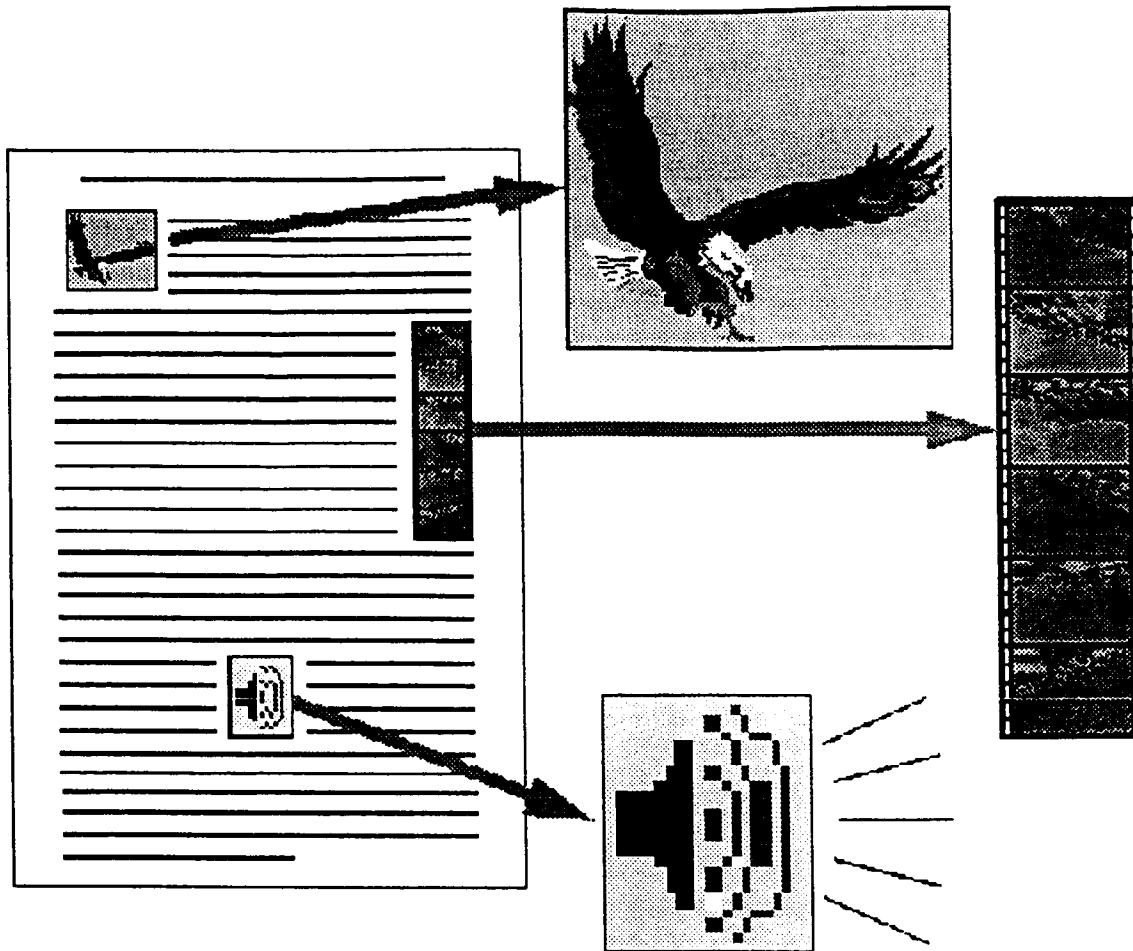
Mosaic provides a more "user friendly" interface to existing Internet services such as Archie, Gopher and WAIS, which allow users to search for and retrieve data from sources throughout the world. Mosaic provides for direct transfer and display of images, motion pictures and sound.

A Hypercard



What is hypertext?

Hypertext is text in a document that is highlighted in some way. When the text is selected, with a single mouse click you will be taken somewhere else in that document or to another related document. We have all probably had some experience with hypertext. PC users have seen hypertext in Microsoft Windows Help--you can click on highlighted text and get more detailed information about that text. Macintosh users first experienced hypertext with the product HyperCard. Many Macintosh products now have hypertext interfaces.



What is hypermedia?

Hypermedia is an extension of hypertext that include pictures, sound, and motion pictures. After a single click on an icon (also called hyperlink -- see below) that represents a picture, sound or motion picture, the object will be displayed, the movie played or the sound produced.

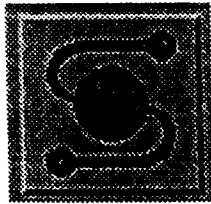
- URL (Uniform Resource Locator) specification (CERN)
 - A Beginner's Guide to URLs
 - URLCurling Up to Universal Resource Locators, by Eric S. Theise
-

Hotlist

Using the hotlist is usually the safest way to be sure that you can come back to interesting information that you have found with Mosaic.

Depending on your version of Mosaic, Hotlist will have its own pull down menu or be found under the "Navigate" pull down menu. If you find a particularly interesting Mosaic screen that you would like to view again, pull down the hotlist menu and add the document to the hotlist. (When you quit Mosaic on the Macintosh, remember to save the changes to the hotlist.) Other WWW browsers may call this same feature "Bookmarks".

Hints



The "S" with a globe in it the NCSA Mosaic symbol and is an indicator that a file transfer is taking place between your computer and a remote computer. This gives you status information on what Mosaic is doing. If a transfer seems to be taking too long or not doing much, you can click on the globe symbol to abort the transfer. (What is too long will depend on the speed of your network connection and how heavily loaded the network is,)

PC Mosaic has a number of problems including being difficult to configure. If you can use X-windows from your PC, it is best to start an X-windows session and use a UNIX version of Mosaic from your PC.

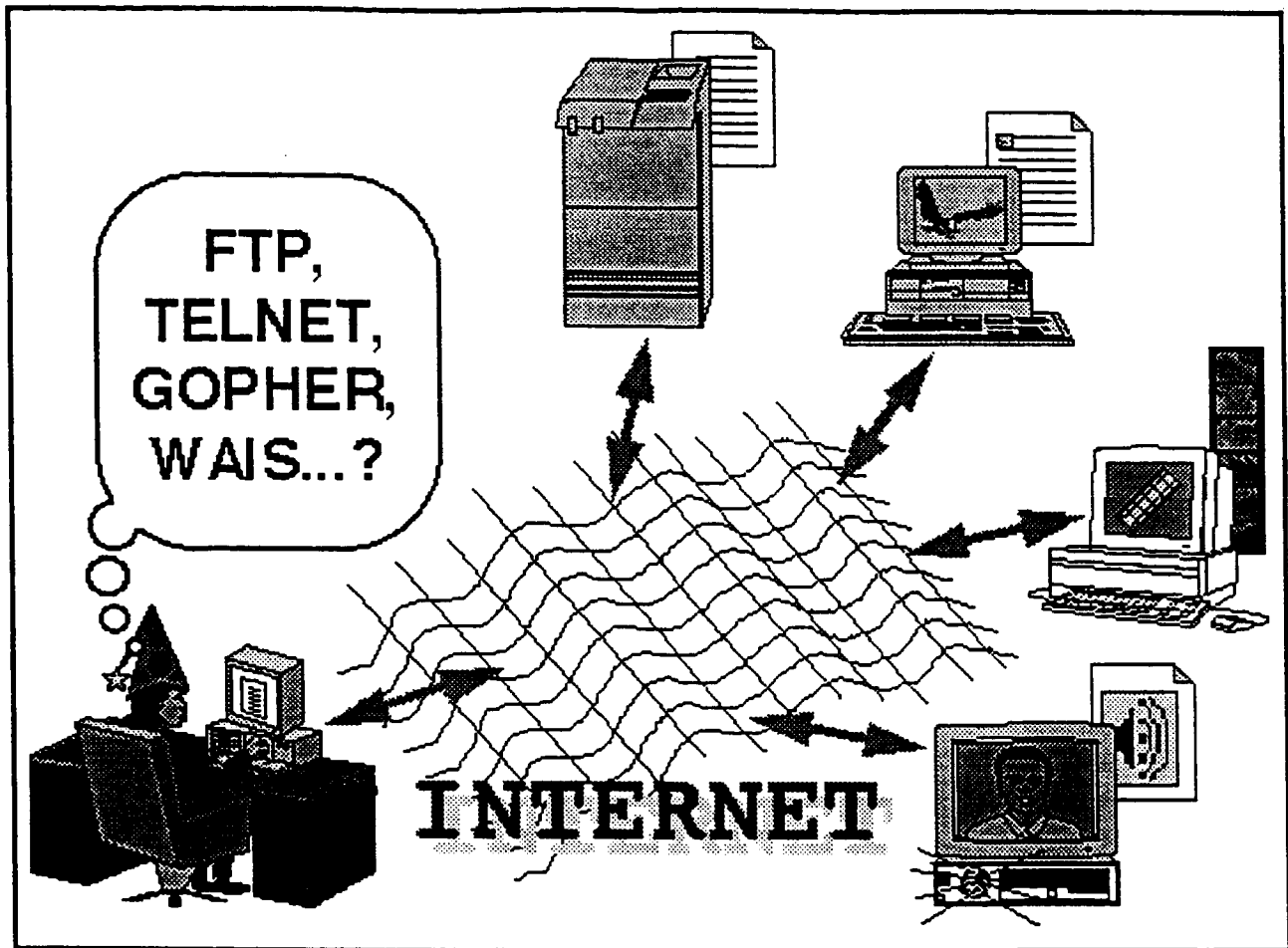
Forms Support is a feature that makes searching for information much easier. The best way to use forms is with X-Mosaic on a UNIX platform. Mosaic for the Macintosh is currently out in version 1.0.3 and does not

support forms. (Mac Mosaic 2.0 Alpha was released on June 10, 1994. At this time it is not known how stable this release is.) PC Mosaic has problems as stated in the paragraph above.

JARGON

Here is some of the jargon you will encounter while using Mosaic and my attempt to explain its meaning:

- Archie - Certain Internet sites maintain lists of the files available at all Internet FTP sites. When you request an Archie search for a given file at one of these servers it responds with a list of all known FTP sites that have the file.
- FAQ - (Frequently Asked Question) Questions that are often asked by new users of the Usenet news services. Many of the Usenet groups create FAQ files to keep network traffic down and avoid repeatedly responding to common questions.
- FTP (File Transfer Protocol) The method used most commonly to transfer files from one computer to another on the Internet. WWW gives FTP a user friendly interface.
- Gopher - A client/server distributed information delivery service. Gopher is like a library where you can browse other libraries' card catalogs and have the material you want automatically sent to you. A deficiency is that one library may have a subject called "Folklore, American" and another may call the same category "Funny Old Stories". (Adapted from The Whole Internet User's Guide & Catalog by Ed Krol)
- HTTP (HyperText Transfer Protocol) A protocol used by the WWW to transfer hypermedia.
- URL - (Uniform Resource Locator) An extended form of file names that locates files and other resources anywhere on the Internet.
- WAIS - (Wide Area Information Service) A client/server distributed information retrieval service. WAIS is like walking into a library with a quote and have the library automatically check out everything that contains it. Think of WAIS databases as private libraries devoted to a particular topic. "In Gopher, you find resources by looking through a



Distributed Hypermedia

Computers have become more sophisticated and able to handle graphical and sound programs. Distributed hypermedia is merely hypermedia (text, sound, picture or movie files) that resides on multiple machines and is accessible via a network.

Hyperlinks/Home Page

Hyperlinks are highlighted text, pictures or symbols in a document that indicate a connection (or link) to other material. When you click on a hyperlink with your mouse you directly access the item that the hyperlink refers to. These documents, pictures, videos, or sounds are files that may reside anywhere on the Internet. Your computer retrieves them as files and opens the proper application to display them as documents, pictures, videos,

or sounds.

A "Home Page" is a hypermedia document that is on the World Wide Web to give information about the posting organization or project. Usually the home page will aim to be eye catching by including a logo for the organization and some picture of the organization's activities. Most home pages also include hyperlinks to other multimedia documents about the organization and related organizations.

HTML

HTML stands for Hypertext Markup Language -- a meta language used to write the hypertext pages of the WWW. The easy to read text that you see on your screen actually comes to you in a format that your computer must then read and format into a form suitable for your display. For example: the title of this section actually looks like:

```
<a name="html"> <h3> HTML </h3> </a>
```

HTML is important in other ways that Donna Roper will cover in her presentation.

[Click here for Donna Roper's presentation on HTML](#)

URL

URL stands for Uniform Resource Locator. A URL may be thought of as an extended filename that lets you find a file anywhere on the Internet. The URL also can have information about what kind of a file it is and other information. All versions of Mosaic have the option "Open URL" under their "File" pull down menu. The URL becomes useful when you see a statement in your email like "The LaRC home page URL is <http://www.larc.nasa.gov/larc.html>" To access the LaRC home page all you need to do is pull down the Mosaic "File" menu and select "Open URL" then type the string "<http://www.larc.nasa.gov/larc.html>" (without the quotes).

sequence of menus until you find something appropriate. WAIS does the same thing, but it does the searching for you. You tell it what you want: it tries to find the material you need." (Adapted from The Whole Internet User's Guide & Catalog by Ed Krol)

Ways to find out more about the WWW:

- The WWW FAQ (Frequently Asked Questions with answers) is very good.
- Read the LaRC Usenet news group "larc.users.mosaic".
- Read the NASA Usenet news group "nasa.infosystems.www".
- Read the Usenet news group "comp.infosystems.www".
- The tutorial at URL <http://matrix.ssd.intel.com:8008/BrownBag/brownBag.html> is excellent.
- A tutorial at URL <http://navigator.jpl.nasa.gov/section314/papers/www-seminar/www-seminar.html> is more technical but still good.

Jim Youngblood (j.r.youngblood@larc.nasa.gov)

356 080

P. 13

110049

N95-16464



Use of World Wide Web and NCSA Mosaic at Langley

CSTC Workshop, H. J. Reid Conference Center, 06/16/94

Michael Nelson, Information Systems Division

<http://blearg.larc.nasa.gov/~mln/cstc/>



Use of World Wide Web and NCSA Mosaic at Langley

- A Brief History of WWW at Langley Research Center
- The Impact of WWW at Langley
- Various Projects That Have Used WWW Successfully
 - Technology Opportunities Showcase
 - Langley Distributed Active Archive Center – EOSDIS
 - Langley Technical Report Server
 - Langley High Performance Computing and Communications K–12 Program
 - COSMIC Replacement
- The Future of WWW at Langley
- What's Next?



A Brief History of World Wide Web (WWW) at Langley

Langley's Leadership Role

- Langley Home Page became public on July 25, 1993
- The initial set of pages were quickly followed by a number of other contributors
- The Langley Home Page is almost a year old
- The Langley Home Page was the **first** NASA center home page
- Why is the '75 Years' logo used?
 - To remind ourselves and others that leading the way is nothing new for Langley
 - And while the technology may be new, the innovative spirit is not

NASA's Leadership Role

- Archie Warnock and Jim Gass of GSFC lead NASA Home Page effort, with input from all of the centers
- Communication through the NASA USENET newsgroup, [nasa.infosystems.www](#)
- The first version of the NASA Home Page became public on September 8, 1993.
- NASA continues to lead federal agencies in deployment and use of WWW
- The NASA Web is a model for grass-roots involvement and inter-agency collaboration



Charateristics of the Langley Web

Architecture of the Langley Web

- Canonical list – “one stop shopping”
- Logically central, physically distributed
- Langley home page is largely a collection of pointers to other WWW servers at Langley and beyond
- Macs, PCs, and UNIX workstations have HTTP servers

The Langley Web Benefits From a Large Number of Contributors

- Over 20 public HTTP servers (plus several others in testing or private)
- Everyone is responsible for maintaining the information they know the most about
- It encourages experimentation
- Everyone is involved with the new information distribution methodology: Its not just "send me an e-mail", its now also "send me the URL"



Impact of World Wide Web at Langley

No Longer the "Best Kept Secret in the Government"

- Statistics not kept until August 27, 1993
- Number of Langley home pages served: **0096410**
- At one point, the Langley home page was the "18th Most Linked to Home Page" (source: a Univ. of Washington Web Crawler)
- 797000+ HTTP connections with main Langley WWW server
- Accesses to the main Langley WWW server (www.larc.nasa.gov)
 - 1700+ Langley Computers
 - 5100+ NASA Computers
 - 62000+ Computers World-Wide
- www.larc.nasa.gov is currently a non-dedicated, SPARCstation IPX, 64 Mbytes memory, 1.5 Gbytes disk

What is the Impact on the WWW Users?

- Move from zero-sum to non-sum information distribution model
- Perhaps most importantly, connecting:
 - People with technologies
 - People with people



Some Langley Projects that have employed WWW

These Projects Have Increased Awareness and/or Usage with WWW

- Technology Opportunities Showcase (TOPS)
- Langley EOSDIS Distributed Active Archive Center
- Langley Technical Report Server (LTRS)
- High Performance Computing and Communications K-12 Program
- COSMIC Replacement

Important Notes About the Above Projects

- Each represent “firsts” in their respective areas
- The projects are accessible through a common interface

A Number of Branches, Divisions, Groups, Teams, and Initiatives Use WWW

- Check the Langley home page for a complete and current list!



Technology Opportunities Showcase

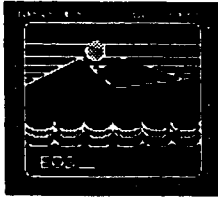


A Diverse and Dynamic N-team Assembled to Construct the TOPS Database

- Number of TOPS home page visitors since 6/01/94: **0000457**
- TOPS builds upon other on-line databases, such as the X.500 phone book information, the Langley Technical Report Server, and existing Langley organization home pages.
- Team members: Kennie Jones (ISD), Jim Fenbert (ASAD), Kathy Stacy (ISD), Gretchen Gottlich (PRMO), Kurt Severance (ISD), Michael Nelson (ISD), Rick Hoff (STID), Dan Axelrad (STID co-op), Chris Matthews (CSC), David Bianco (CSC), Tricia Smith (ISD)
- Others latered contributed tours, reports and other information
- Features: all data sheets; keyword searching; photographs; "clickable" TOPS floorplan; automated metrics; and on-line requests for more information forms
- POC: Kennie Jones, K.H.JONES@LaRC.NASA.GOV, 864-6720
- <http://www.larc.nasa.gov/tops/tops.html>



Langley Distributed Active Archive Center (DAAC)

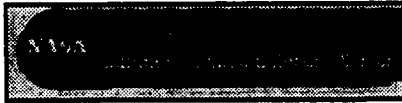


A Component of the Earth Observing System Data Information System (EOSDIS)

- The Langley DAAC uses a home page to:
 - Increase awareness of the Langley DAAC
 - Provide various documentation sets
 - Provide user services information
 - Launch the innovative *Langley DAAC Data Ordering System X Window System/Motif client*
- Some projects currently served with DAAC: ERBE, SAGE, FIRE, SRB, ISCCP
- DAAC use of WWW has enabled several hundred more data set transfers
- POC: Roy Dunkum, R.C.DUNKUM@LaRC.NASA.GOV, 864-6589
- <http://eosdis.larc.nasa.gov/>



The Langley Technical Report Server (LTRS)

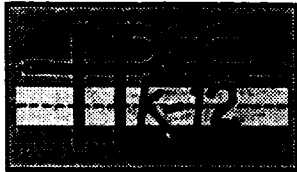


LTRS is an Experimental Report Distribution Project

- Distributes “**unclassified, unlimited**” technical reports and papers
- Began January 1993 as an Anonymous FTP server only – (WAIS searching adding shortly thereafter)
- In the first 6 months (1/93 – 7/93), 2400+ reports distributed (pre- WWW)
- WWW enabled integrated searching and retrieving in October 1993
- As of 6/94, 10000+ reports distributed
- WWW provides a more intuitive and friendly interface to LTRS
- LTRS concept is being replicated across NASA via the NASA Technical Report Server (NTRS)
- RPPB (former – Technical Editing Br.) provides formal publications; others are contributed by the authors
- LTRS team members: Michael Nelson (ISD), Gretchen Gottlich (PRMO), David Bianco (CSC)
- POC: Michael Nelson, M.L.NELSON@LaRC.NASA.GOV, 864-8511
- <http://techreports.larc.nasa.gov/ltrs/ltrs.html>



The Langley High Performance Computing and Communications K-12 Program



The Langley HPCCP K-12 Program is Active!

- Five area high schools are currently class C registered networks on the internet (e.g., *patriot.denbigh.nn.k12.va.us* is a valid Internet address)
- Three new schools are scheduled to be online this fall
- Each school currently receives its network connection from Langley over standard phone lines, and has a collection of donated Sun UNIX workstations and Apple Macintoshes
- The teachers are learning about computation, and integrating it into the curriculum
- All volunteer effort: Gary Warren (FMAD), Leon Clancy (ICASE), Kelvin Edwards (AS&M), plus others

The Langley HPCCP K-12 Program Has Received Broad National Recognition

- The Langley K-12 program is a fixture on educational WWW pages
- The Langley host machine for K-12 has registered over 20000 individual file accesses
- POC: Gary Warren, G.P.WARREN@LaRC.NASA.GOV, 864-2162
- <http://k12mac.larc.nasa.gov/hpcck12home.html>



A Langley COSMIC Replacement is Planned

The WWW is a Natural Medium for Langley Computer Program Distribution

- A prototype is planned for this summer
- All non-sensitive, classified, or controlled programs would be available for free and open distribution
- Inspired by Oak Ridge National Lab's Netlib, which processed over 1.8 million requests in 1993
- Implemented by a TAG-lead N-team
- Will build upon work already done with the Langley Technical Report Server
- Sample codes are sought
- POC: Dan Sydow, P.D.SYDOW@LaRC.NASA.GOV, 864-3180



The Future of WWW and Mosaic at Langley?

Complete the Langley Web

- Currently, only a portion of Langley's activities are represented
- Everyone should be able to maintain at least minimal information about their organization or project
- Automated inclusion of on-line organization trees, functional statements, etc.

Further in the Future...

- A wider choice of WWW clients, both commercial and freeware
 - Mosaic has been licensed to several companies for commercial development
 - NCSA Mosaic will continue to develop and remain freely available
- Tighter integration of all WWW documents
- Better searching tools
- Better authoring and data management tools
- Sophisticated "Knowledge Robots" that search, retrieve, and filter various information sources according to personal preferences



Concluding Remarks

The World Wide Web and NCSA Mosaic Have Changed the How Langley Does Business

- Langley and NASA lead in the adoption of WWW technology to accomplish our Mission
- Several projects and programs have already enjoyed tremendous success using WWW
- WWW is now an integral tool for technology transfer both out of and into Langley
- Langley is no longer a "secret"; and less and less means **Air Force** or **CIA**
- Langley must continue to increase the number of its WWW providers and users

Being on the WWW is Simple, Effective, and Fun!

- Some instructions are available from the Langley home page
- Find a branch, project or other home page that you like and adapt it
- Come to the Internet Fair, June 28, H. J. Reid Conference Center, Langley Research Center, 8am – 3pm for more information
- And the next presentation will explain how to get started if you can't wait!

356 081

110050

N95- 16465

How To Use the WWW To Distribute STI

by Donna G. Roper

P. 10

This presentation explains how to use the World Wide Web (WWW) to distribute your scientific and technical information (STI) as hypermedia. WWW clients and servers use the HyperText Transfer Protocol (HTTP) to transfer hypermedia documents, that is, documents containing links to other text, graphics, video, and sound. The standard language for these documents is the HyperText Markup Language (HTML). HTML documents are simply text files with formatting codes that contain layout information and hyperlinks. To make your scientific and technical information available to the WWW as hypermedia documents, you must learn how to create HTML documents and make them available on an HTTP server. You can create HTML documents with any text editor or with one of the publicly available HTML editors or converters. You can also use HTML to include links to image formats such as XBM, GIF, TIFF, JPEG, MPEG. Most of the information that you need to get started is available on the Internet. This presentation is available on-line. The URL is <http://sti.larc.nasa.gov/demos/workshop/introtex.html>

Using the WWW for STI Allows Users To

- Refer back to equations, figures, and text in previous sections
- Access references that are available on-line
- Attach personal, group, or public annotations to documents
- Download figures for manipulation or inclusion in other reports
- Download computer codes, programs, and documentation
- Access simulation models, data files, and videos
- Browse files in HDF (Hierarchical Data Format), a machine-independent file format that allows arbitrary grouping and annotation of heterogeneous data elements.
- Send scientific data in a hypermedia document across the network for graphical and statistical inspection and analysis on programs such as
 - Collage, NCSA's synchronous collaboration tool for scientific data analysis and manipulation.
 - Polyview, NCSA's collaborative tool for three-dimensional geometric and polygonal data analysis.
 - Data Management Facility (DMF), NCSA's scientific data management and archival system.

Distributing STI on the WWW as Hypermedia

- Learn How To Create HTML Documents.
- Make Documents Available on an HTTP Server.

Hypermedia Documents Contain Links To

- **Text**
(7-bit ASCII)
 - **Graphics**
(e.g., Graphs, Photos, Line Drawings)
 - **Video**
(e.g., Crack Propagation or Air Flow Over Wing Configuration)
 - **Sound**
(e.g., Engine Noise, Narration)
-

HyperText Transfer Protocol (HTTP)

HTTP is a stateless search, retrieval, and manipulation protocol with the speed necessary for a distributed hypermedia information system.

These HTTP Servers Are Available on the Internet

- NCSA httpd
- BSDI Plexus
- GN (a gopher/http server from NWU)
- CERN HTTP server
- MacHTTP - a Macintosh HTTP server
- serweb - Windows 3.1/NT HTTP server (requires winsock)
- HTTPS - Windows NT HTTP server (for PCs and Alphas)
- NCSA httpd for Windows

Your system administrator should be able to help you set up the http server. If not, contact m.l.nelson@larc.nasa.gov about serving files from www.larc.nasa.gov

HyperText MarkUp Language (HTML)

HTML documents are 7-bit ASCII files with formatting codes that contain layout information and hyperlinks to text, graphics, video, and sound.

How To Create HTML Documents

- Create HTML Documents With any Text or [HTML Editor](#)
 - Create HTML Documents With a Word Processor and Export as ASCII
 - Create Documents With a Word Processor and [Convert to HTML](#)
-

Sample HTML Document

HTML References

- [A Beginner's Guide to HTML](#)
 - [Crash Course on Writing Documents for the Web](#)
 - [Elements of HTML Style](#)
 - [HTML Tutorial](#)
-

Tips For Writing HTML

- Save As HTML Option
- Open Local Option

Sample HTML Document

```
<h1> Heading Level One </h1>
```

This text is a sample paragraph. Paragraphs must be separated with the html paragraph tag because

blank lines and tabs are ignored.

```
<p>
```

This text is another sample paragraph. You can use html tags to display *italic text*,

```
<h2> Heading Level Two </h2>
```

This text contains an unordered list.

```
<ul>
```

```
<li> Item 1
```

```
<li> Item 2
```

```
</ul>
```

Heading Level One

This text is a sample paragraph. Paragraphs must be separated with the html paragraph tag because blank lines and tabs are ignored.

This text is another sample paragraph. You can use html tags to display *italic text* and **bold text**.

Heading Level Two

This text contains an unordered list.

- Item 1
 - Item 2
-

HTML Link To Another Document

You can link regions of text or images to another document or image as well as to a specific section in a document. Here is a hypertext link (*called an anchor*) to the next document.

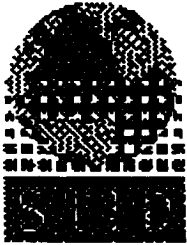
Here is the HTML tag:

```
<a href="image.html"> next document </a>
```

Mosaic Can Display Inline Images in Two Formats

- XBM (X Bitmap)
- GIF (Graphic Image Format)

For example, here is the logo for our division



Here is the HTML tag:

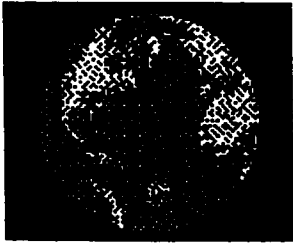
```

```

Mosaic Can Open External Images in These Formats

- XBM (X Bitmap)
- GIF (Graphic Image Format)
- HDF (Hierarchical Data Format)
- PS (PostScript Format)
- TIFF (Tagged Image File Format)
- JPEG (Joint Photographics Expert Group)
- MPEG (Motion Pictures Expert Group)
- Any Format For Which You Have a Viewer

Here is an inlined image (thumbnail) with a hypertext link to a higher resolution photo in JPEG format.



Here is the HTML tag:

```
<a href="http://sti.larc.nasa.gov/photos/photo2.jpg">
```

```
 </a>
```

Displaying Scientific Equations

Display Scientific Equations As Inline Images

Here is an example of an equation in XBM format:

$$\beta = \tan^{-1} \left(\frac{M_{2n}}{M_{2p}} \right) = \tan^{-1} \left(\frac{M_{2n}}{(T_1/T_2)^{1/2} M_1 \sin A} \right) \quad (14 K)$$

Here is an example of the same equation in GIF format:

$$\beta = \tan^{-1} \left(\frac{M_{2n}}{M_{2p}} \right) = \tan^{-1} \left(\frac{M_{2n}}{(T_1/T_2)^{1/2} M_1 \sin A} \right) \quad (5 K)$$

A separate HTTP connection must be made to retrieve each inline image, which is stored in a separate file. Thus, documents with multiple images take longer to download and require more storage for the document elements.

Sample Documents With 21 Equations

- Equations converted to X Bitmaps: 229 KB total: 16 seconds
- Equations converted to GIFs: 13.8 KB total: 11 seconds

(Source: *"Thoughts On Scientific HTML Documents"* by M.C. Grant from Stanford University.)

Problems With Displaying Scientific Information

- HTML Does Not Support Greek & Mathematical Symbols.
- Equations Are Stored in Multiple Files.
- Some HTML Converters Ignore Equations.
- Equations Are Difficult To Align With Text.
- Equations Are Not Scaled To Match Text.

• Superscripts & Subscripts Are Not Supported.

• Tables Are Difficult To Format.

The next version of HTML (*called HTML+*) will address some of these issues.

Sample Table

Table 6. Parallel Golden Block Method

No of Proc.	No. of Points	Time (sec)	Speedup for	
			PGB	GS
1	2	0.355	1.00	1.00
1	12	0.540	1.00	0.66
2	12	0.277	1.95	1.28
3	12	0.187	2.89	1.90
4	12	0.144	3.76	2.47

SESSION 7 Graphics and Image Processing

Chaired by

David C. Banks

- 7.1 **Image Tools for UNIX - David Banks**
- 7.2 **From Computer Images To Video Presentation: Enhancing Technology Transfer - Sheri Beam**
- 7.3 **Data Visualization and Animation Lab (DVAL) Overview - Bill Von Ofenheim , Kathy Stacy**
- 7.4 **Data Visualization and Animation Lab Applications - Kurt Severance and Mike Weisenborn**

Image Tools for UNIX

David Banks, ICASE

There are many tools available for digital image processing in the UNIX environment. This talk features two tools that are simple and useful: xv and pbmplus.

The xv image viewer runs under the X window system. It reads images in a number of different file formats and writes them out in different formats. The view area supports a pop-up control panel (activated by pressing the right-most mouse button). This control panel has a file selector, a menu bar, and several buttons at the bottom. The "Algorithms" menu item lets you blur an image. Why would you want to blur an image? One reason to blur is that you might wish to shrink the image. Without blurring first, a "shrink" operation generally obliterates any of the fine details that are in the full-size image. The bottom buttons let you flip, crop, and resize an image.

The "xv" control panel can also activate the Color Editor. The Color Editor displays the image's colormap (if it has one). You can select individual elements of the colormap and change their color. This is especially useful if the image has a solid-color background that you wish to change. The Color Editor also applies global changes to the image color. These changes include re-mapping the hues, setting the white-balance, setting the color saturation, and changing the overall intensity mapping. These operations are useful for preparing an image to be printed on a medium that has special color characteristics. As a simple example, color monitors have a wide range of brightness characteristics. An image can be adjusted to match the settings on different display devices.

The xv image viewer is available from the internet at various ftp sites. A postscript manual is available on the world wide web (WWW). It describes a licensing arrangement with the author (at \$25 per machine), but his phone number is no longer valid and neither is his e-mail address. Previous versions of the viewer (before version 3.0) did not mention a license.

The "pbmplus" package is a set of tools designed to perform image processing jobs from within a UNIX shell. The acronym "pbm" stands for "portable bitmap." A bitmap is a straightforward encoding of a black-and-white image. In a pbm file, zeros and ones represent black and white dots in a rectangular array. A portable graymap uses a larger set of values to encode different levels of gray from black to white. A portable pixmap uses triples of integers to encode the red, green, and blue components of each pixel in an image. A portable anymap encodes any prescribed number of integer values for each pixel.

Like "xv", the pbm tools can convert images from and to many different file formats. There are more than 100 individual executable programs in the toolkit; most of them convert images from one format to another. The "pbm" tools do not provide a stand-alone interactive program like "xv" does. Instead they act as filters, taking images as input and producing images as output.

The source code and man pages for "pbmplus" are available by ftp. This software is in the public domain.

Image Tools For UNIX

David Banks, ICASE

The xv Image Viewer

Converts file formats

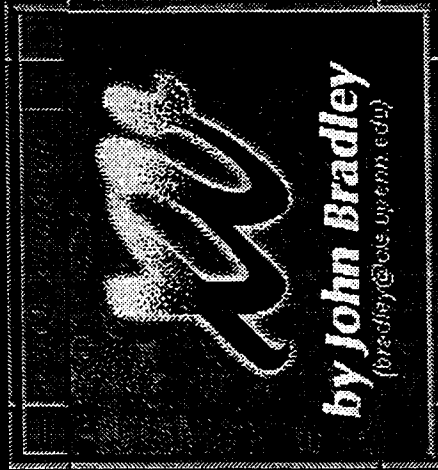
Edits colors

Performs image operations

xv Converts File Formats

gif
pbm
x11
sun
rgb
jpeg
tiff

gif
pbm
x11
sun
ps
rgb
jpeg
tiff



xv Edits Colors

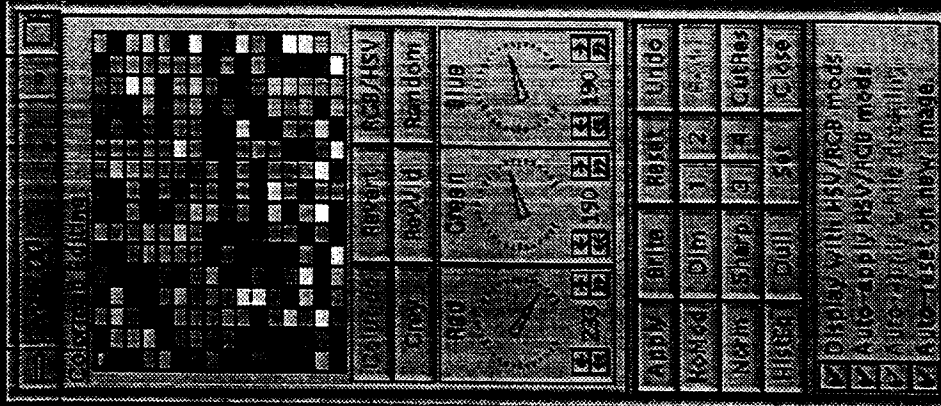
Color map



Color value



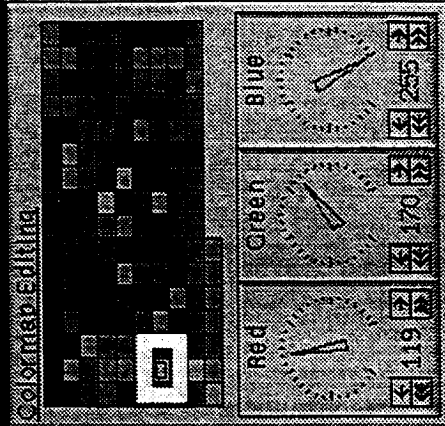
Color ops



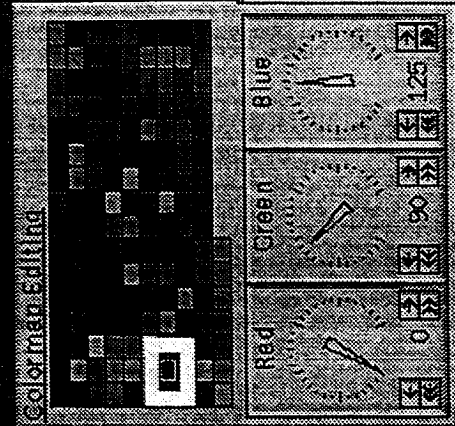
Picking Colors



119,
170,
255



0,
90,
125



xv Edits Colors

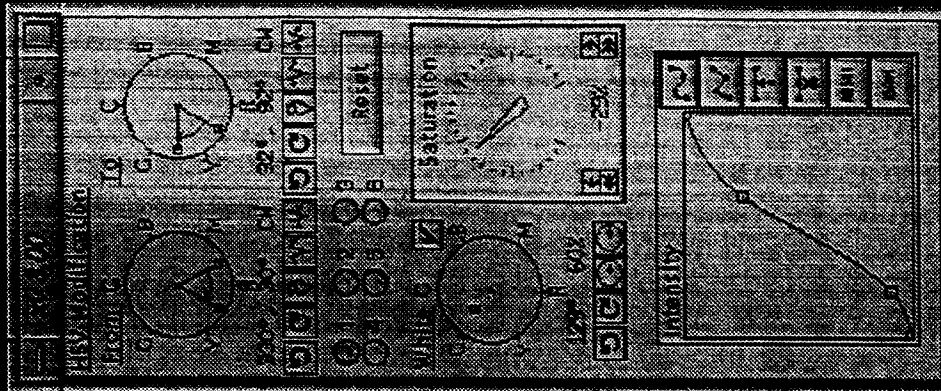
Hue shift



Balance & saturation

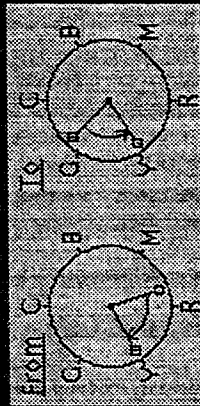


Intensity

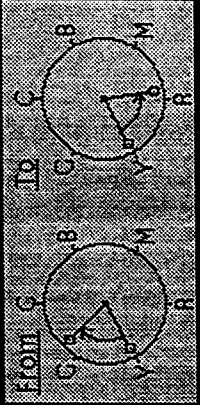


Shifting the Hue

Red to Green



Green to Red



Balance, Saturation, Intensity



Unsaturated



Highlights reduced

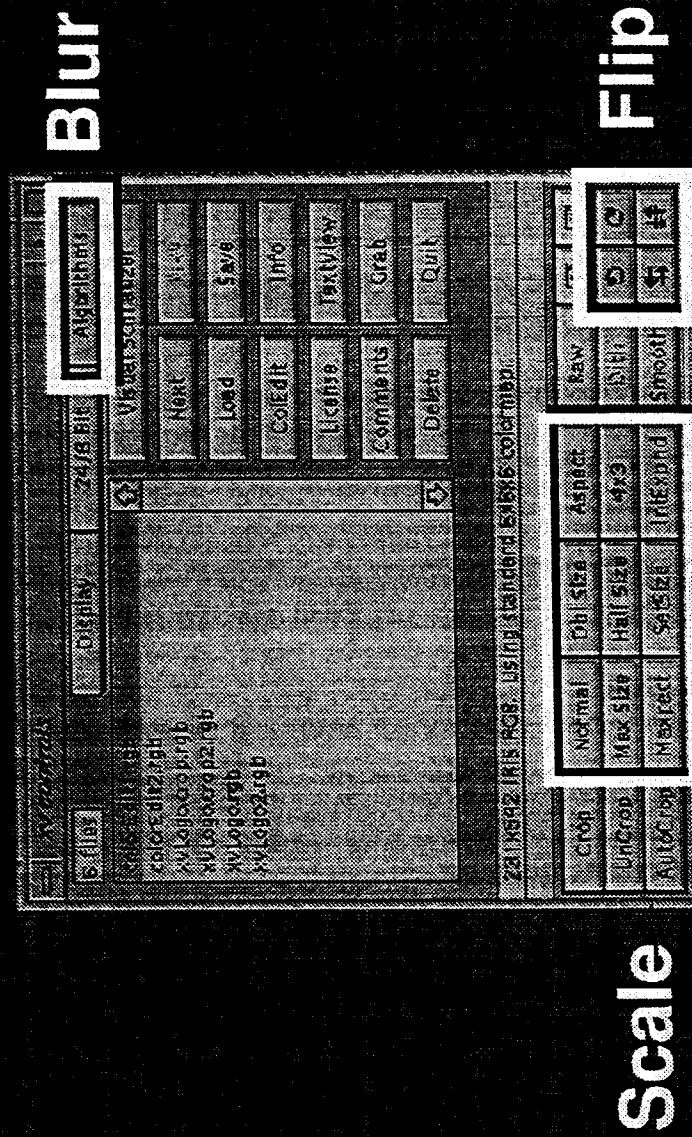


Original



White balance = sepia

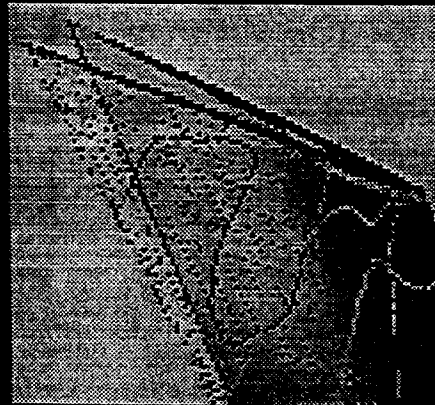
xv Performs Image Ops



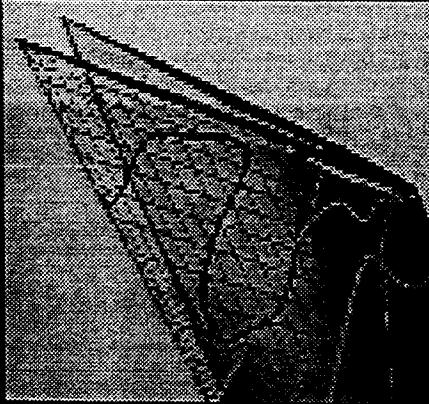
Why Blur an Image?

Shrinking obliterates fine details

Blurring extends fine details



**reduce
only**



**blur &
reduce**

xv From the Net

Available via ftp

[ftp.x.org/contrib](ftp://ftp.x.org/contrib)

Postscript manual on WWW

<http://www.cm.cf.ac.uk/Applications/xvdocs.html>

pbmplus

pbm = "portable bitmap"

Converts file formats

Performs image operations

Evolution of pbmplus

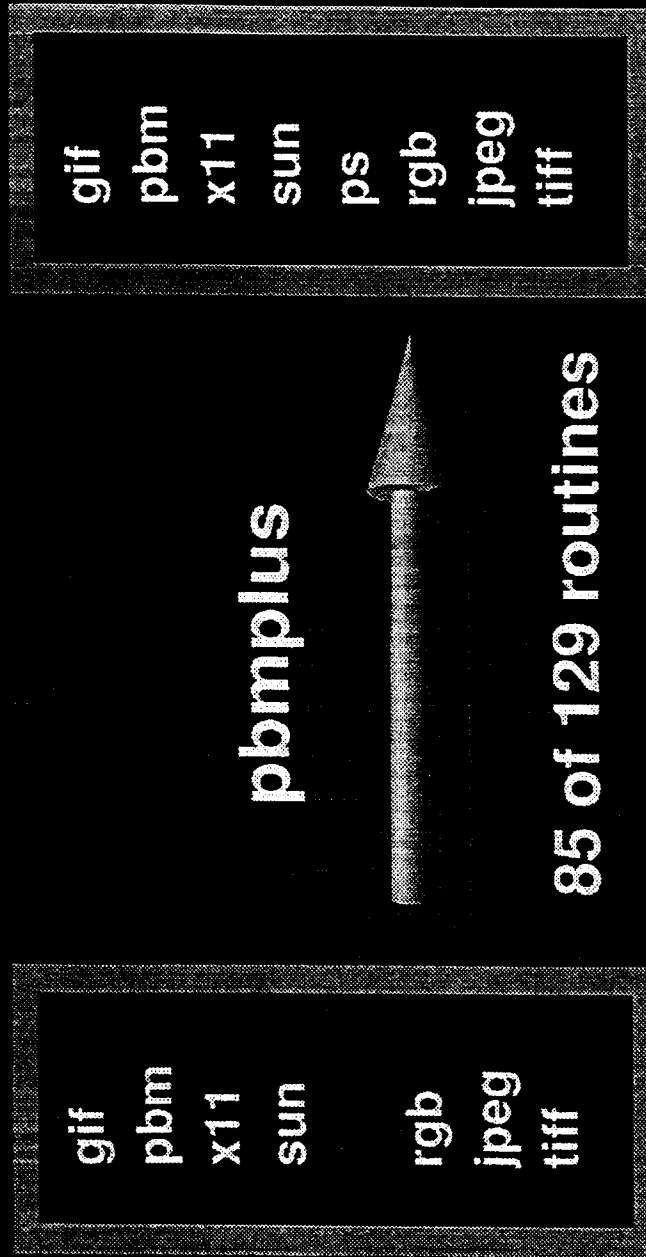
pbm = "portable bitmap"

pgm = "portable graymap"

ppm = "portable pixmap"

pnm = "portable anymap"

pbmplus Converts Formats





p?m Text Format

p2 *This designates a pgm file*
4 3 *The image's x and y dimensions*
255 *The maximum intensity*

```
g g g  
128 128 128 128  
128 128 128 128  
128 128 128 128
```

p3m Text Format

p3 *This designates a ppm file*
4 3 *The image's x and y dimensions*
255 *The maximum intensity*

r	g	b	r	g	b	r	g	b
0	0	255	0	0	255	0	0	255
0	0	255	0	0	255	0	0	255
0	0	255	0	0	255	0	0	255

Examples

pnmsmooth big.ppm > blur.ppm

blur a color image ...

pnmscale 0.5 blur.ppm > small.ppm

... shrink it ...

ppmtopgm small.ppm > small.pgm

... make it grey ...

ppmtogif small.pgm > small.gif

... and convert it to gif

pbmpplus From the Net

Available via ftp

[ftp. ee. lbl. gov](ftp://ee.lbl.gov)

[ftp. x. org/ contrib](ftp://x.org/contrib)

[gatekeeper. dec. com/ pub/ graphics](ftp://gatekeeper.dec.com/pub/graphics)

Tar file includes man pages

From Computer Images to Video Presentation:
Enhancing Technology Transfer

Sheri Beam
Hampton University

P. 6

With NASA placing increased emphasis on transferring technology to outside industry, NASA researchers need to evaluate many aspects of their efforts in this regard. Often it may seem like too much self-promotion to many researchers. However, they should first take a long, hard look at how industry promotes itself.

Industry has been in the video production business for years. Upon a close examination of sales, advertising, public relations and training, video is used everywhere. In fact, in many cases, the quantity of outside production has often dictated the need for many industries to build their own in-house production facilities. In marketing themselves and their products through the use of videotape, industries have been educated by film and video professionals to expect a certain level of quality and sophistication. In addition, industry professionals are familiar with current television programs, like NOVA, which reinforce what they know about the state of the art of video. Therefore, anyone who wants to do business with them, must meet on a level playing field by emulating these very same video production standards.

Today the most typical presentation method at NASA is through the use of vu-graphs (overhead transparencies), which can be effective for text or static presentations. However, for dynamic, full-blown color and sound presentations, the best method is videotape. In fact, it is frequently more convenient, because of portability and the availability of viewing equipment. Due to the nature of its ease of operation, both in the recording and playback, coupled with the fact that viewing television is passive, many people suffer from the misconception that creating a video production is also a simple and passive activity.

Although a NASA researcher may not use the same approaches to create a computer-generated presentation as an entertainment program, some aspects are essential for both if they will eventually be viewed on a video monitor. The intended audience must be identified, as this will help to determine the level of technical content, as well as the length of the presentation. A good presentation can be compared to a good story. It has a beginning, a middle and an end. For technology transfer purposes, a researcher should try to introduce the research images, give the details of the research, and review the images and information presented.

When creating the computer images for the presentation, a major consideration is the viewing environment. The size of the space, plus the size of the monitor screen, plus the number of people viewing the presentation should determine the number of screens necessary for an effective presentation. Since the most common videotape used in the United States is still VHS (1/2") NTSC (National Television Standards Committee) format, the computer images will have to meet certain requirements in order to maintain the possible best quality through the transfer process. Although the computer has the ability to accurately reproduce a multitude of colors in intense saturation levels, the video monitor has much more limited capabilities. Primary colors, often the first choice of the researcher, are particularly difficult to reproduce.

Screen composition is another important consideration. At present, video monitors typically have a three by four screen ratio. With this basic horizontal format in mind, a researcher can create more aesthetically pleasing images by following established principles from great artists, including foreground, middle ground, background, balance and lighting. If the image is animated, employing accurate simulation to reality, including initiation, direction, smoothness, and completion are important criteria to follow.

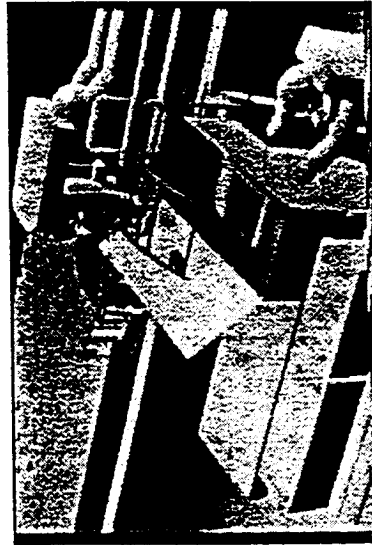
It may take many hours to create and render just one frame of the image, but real time video runs at 30 frames per second. A researcher needs to keep this in mind when generating animations and determining how long they should run. One that runs slowly due to a lack of frames (e. g. 6 fps) will also not run smoothly. It is better to play it more frequently at a faster rate.

Since a researcher may not always be available for the actual presentation, professionally edited audio narration on the videotape can make it an effective stand-alone product. Writing the script before editing the images facilitates matching picture and sound. Although a technical paper of the research may have been previously published, it will have been written for the eye and not for the ear. For this reason, a collaboration with someone trained in writing for broadcast is necessary.

The researcher and the video professional combine to form a unique team, blending the scientific with the aesthetic, where all the necessary detailed steps take shape in a creative concept. This concept ultimately becomes a video presentation at the level of quality expected by outside industry.

**From Computer Images to Video Presentation:
Enhancing Technology Transfer**

"industry already knows ..."



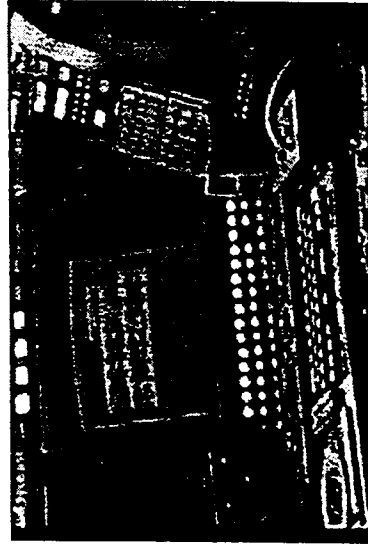
Lockheed



NOVA

From Computer Images to Video Presentation: Enhancing Technology Transfer

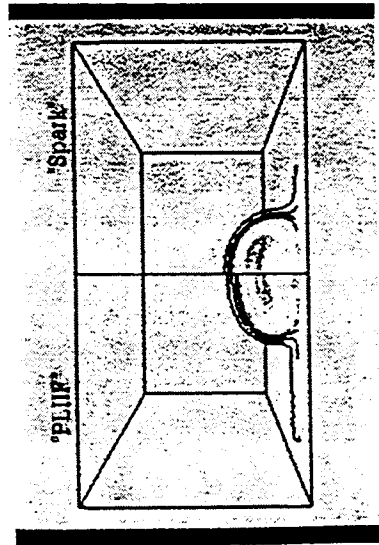
"but this isn't entertainment ..."



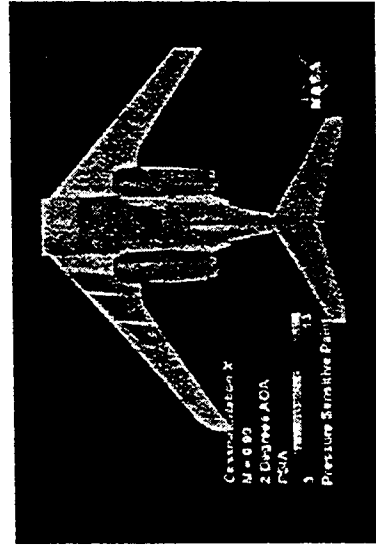
"Alien"

From Computer Images to Video Presentation: Enhancing Technology Transfer

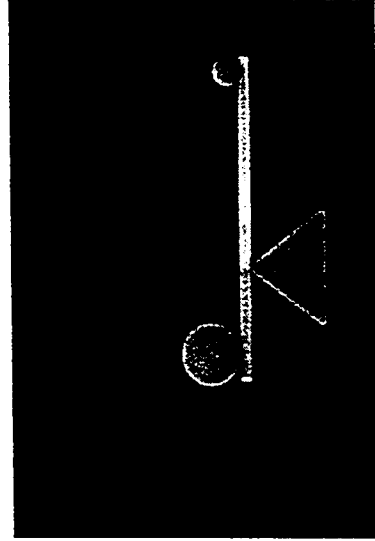
- "audience"
- "tell a story"
- "MOTION"
- "color"
- "sound"



motion



color



composition

**From Computer Images to Video Presentation:
Enhancing Technology Transfer**

"the right and left work together ..."



356125

110053

N95-16468

The Role of Computers in LaRC R&D
Graphics and Image Processing Session
June 16, 1994

P. 20

Data Visualization and Animation (DVAL) Overview

Presented by Kathy Stacy and Bill von Ofenheim

Abstract:

The Data Visualization and Animation Lab is an open shop facility created and supported by the Scientific Applications Branch of the Information Systems Division. The DVAL is located in Building 1268, Room 1101A. An experienced team of visualization experts is available to help researchers import, visualize, and interpret data derived from a wide variety of sources including in-flight experiments, wind tunnel tests, computer simulations, and atmospheric studies.

The general capabilities of the DVAL include digital image processing, 3-D interactive computer graphics, data visualization and analysis, video-rate acquisition and processing of video images, photo-realistic modeling and animation, video reports generation, and color hardcopies. The hardware resources of the facility cover a variety of computer platforms including Sun workstations, SGI workstations, PCs, and Macs. The Video Image Processing System (VIPS) is a specialized system designed for post-processing of images recorded to videotape. The system supports the common video formats used at the Center, including VHS, S-VHS, U-Matic, U-Matic SP, and Betacam. The most common application of VIPS is the processing and analysis of video images produced by wind tunnel or in-flight flow visualization experiments. The system allows for video-rate (or 30 frames per second) digitization, processing, storage and retrieval of video frames. The real-time digital disk can store up to eight minutes of digital image data. The video-rate processing includes frame averaging, running averages, frame-by-frame subtraction, pseudocoloring, and spatial convolutions with a kernel size up to eight by eight. The Scientific Visualization System (SVS) is another specialized system for generating broadcast quality video productions. Hardware resources also include a film scanner and flatbed scanner for image input, and graphics hardcopy devices for image output. The software resources include major commercial visualization packages such as PV-WAVE, KB-Vision, SGI Explorer, WAVEFRONT, TECPLOT, Mathematica, and Fieldview, as well as public domain packages and software packages developed in-house.

A sample application which utilizes most of the capabilities of the DVAL is the F-106B Leading-Edge Flow Visualization Experiment. The original data from this experiment were vapor screen images recorded on black and white VHS videotape. Select frames from the videotape were digitized in DVAL using the VIPS system. The 2-D digital images could then be enhanced, and vortex core locations could be located, using PV-WAVE software. A geometric mapping model was developed to accurately map 2-D vapor screen images into 3-D space. The Flow Analysis Software Toolkit (FAST) was used to interactively visualize the 3-D vapor screen image data along with the numerical surface geometry of the F-106B. Computer animations were generated using FAST, and a broadcast quality video containing these animations was produced on the SVS.

One component of DVAL is the Scientific Visualization System (SVS) which

consists of a state-of-the-art digital video editing suite for creating broadcast-quality videos from computer-generated results. These videos are used for analysis, presentation, and dissemination. Video helps the analysis process by providing real-time playback of images which may take hours to create thereby allowing researchers to get a better understanding of their time-dependent results. Video is also a highly portable and universal media for presenting dynamic data at conferences and meetings. Lastly video is an effective mechanism for abetting the technology transfer process by virtue of its inexpensive and self-contained nature.

The philosophy behind the Scientific Visualization System is the preservation of the original image quality. This is accomplished by using digital component video equipment. Digital component video is compatible with digital computers and digital networks so image data suffers no loss during transmission. Also editing and special effects are performed digitally so the integrity of the original image is always maintained.

There are three phases to the video creation process: 1) Pre-Production, 2) Production, and 3) Post-Production. The pre-production phases involves creating a storyboard, writing a script, narrating the script, and adding music. Not all of these steps are required for each video but at minimum each video should start from a storyboard. The production phase involves creation of the images or animations using existing packages (e.g. FAST, Wavefront, and TECPLOT) or special purpose codes written by SVS personnel. The created images/animations are then transferred to SVS either digitally using LARCNET or via analog means such as SVS's transportable laser disk recording system. Video tapes created using a video camera such as used in wind tunnel and in-flight experiments are another means for production. Finally, the video is edited together in the post-production phase using editing techniques (e.g. fade, dissolve, wipe, etc.), special effects (e.g. warps, split screens, layering, etc.), title generation, paint, and graphics.

**1994 Workshop
The Role Of Computers in LaRC R&D
Graphics and Image Processing Session
June 16, 1994**

DATA VISUALIZATION & ANIMATION LAB (DVAL) OVERVIEW

**Presented by:
Kathy Stacy
Bill von Ofenheim**

Background

- Open shop scientific visualization facility
- Located in Building 1268, Room 1101A
- Created and supported by the Scientific Applications Branch of ISD
- Objectives:
 - 1) provide and maintain a state-of-the-art scientific visualization capability at LaRC
 - 2) foster partnerships with LaRC researchers to apply visualization tools and techniques to enhance and enable science investigations

General Capabilities

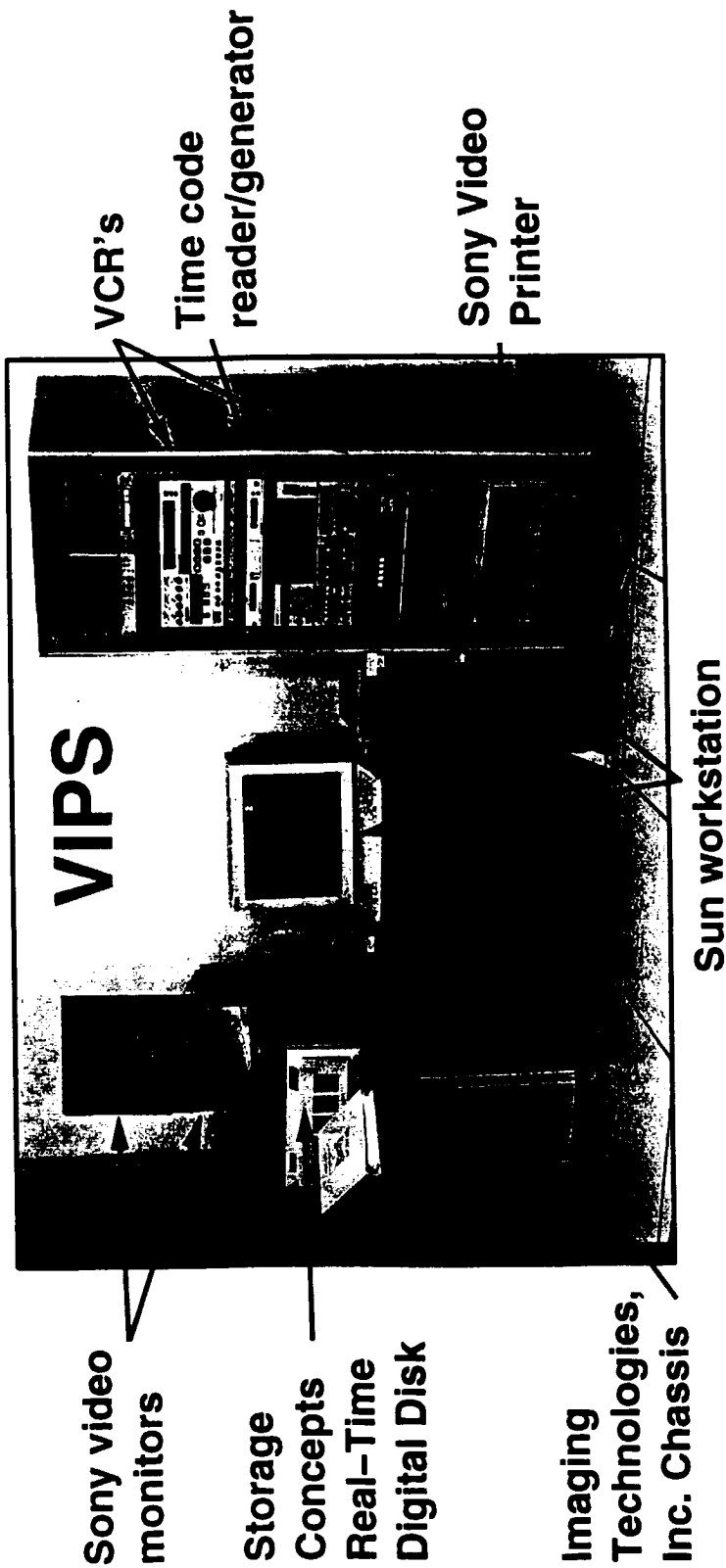
- **Digital Image Processing**
- **3-D Interactive Computer Graphics**
- **Data Visualization and Analysis**
- **Video-rate acquisition and processing of video images**
- **Photo-realistic Modeling and Animation**
- **Video reports generation**
- **Color Hardcopies**

Resources

- **Hardware Resources:**
Sun workstations, SGI workstations, PC, Mac
- Video Image Processing System (VIPS)**
- Scientific Visualization System (SVS)**
- Film scanner and flatbed scanner**
- Access to Production Graphics Hardcopy Devices**
- **Software Resources:**
Major commercial packages including PV~WAVE, KB-Vision, SGI Explorer, WAVEFRONT, TECPLOT, Mathematica, Fieldview
- Public Domain packages including KHOROS, Xv, Xloadimage**
- In-house packages including ILLUME, Blobtool, Thrtool**

Video Image Processing System

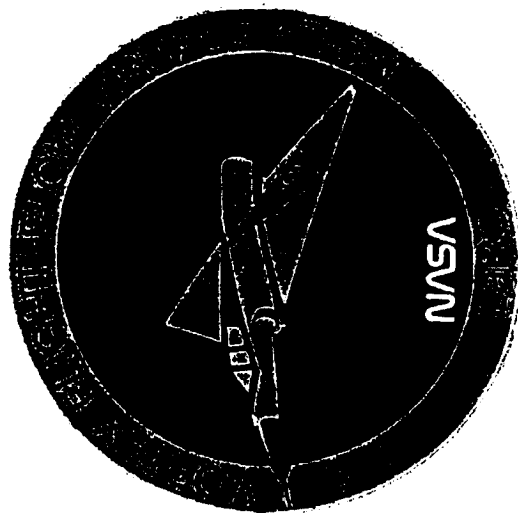
- Designed for post-processing of images recorded to videotape
- Common applications include processing of wind tunnel or in-flight flow visualization experiments recorded on videotape
- Video-rate digitization, processing, storage, and retrieval



DVAL OVERVIEW

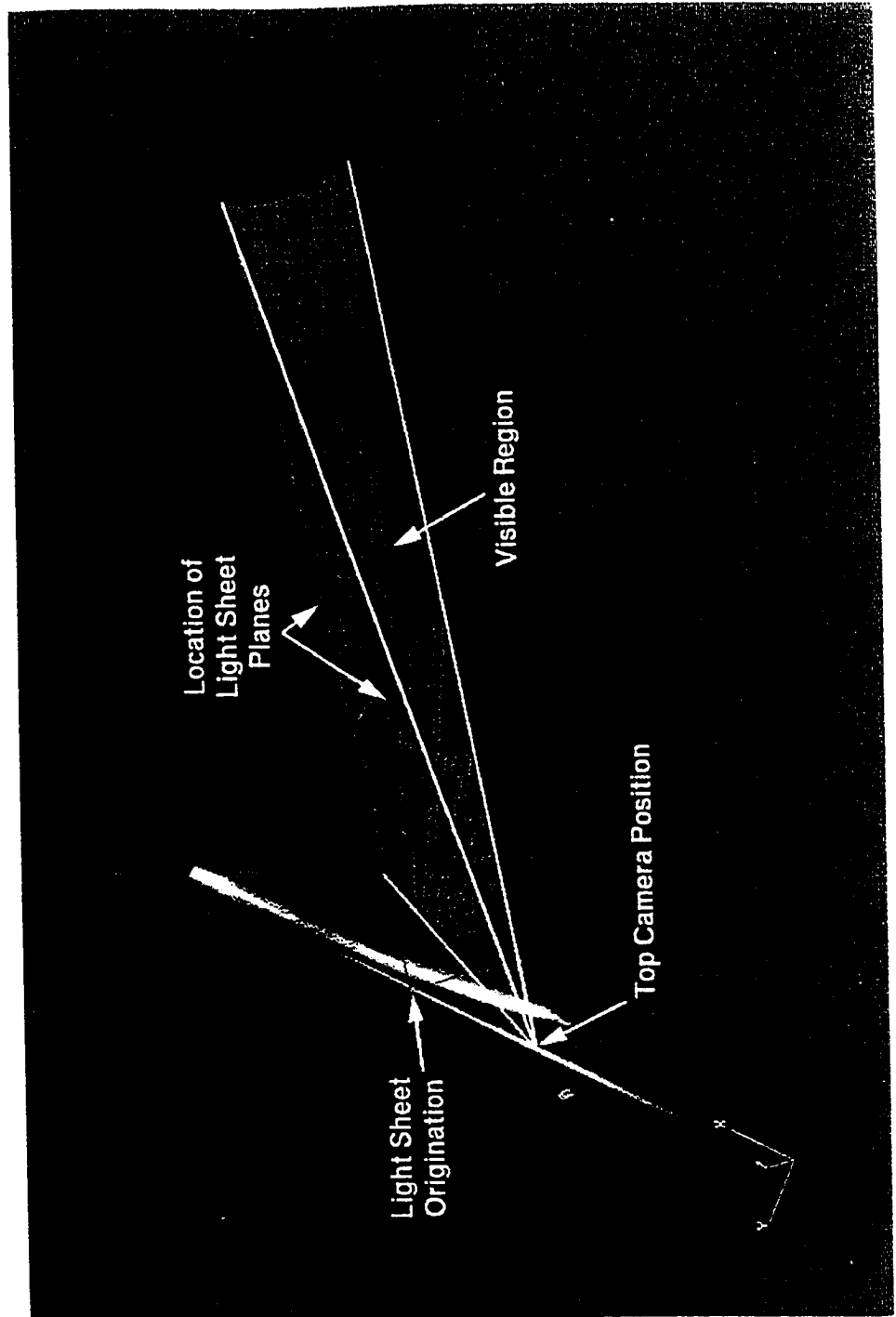
Sample Application

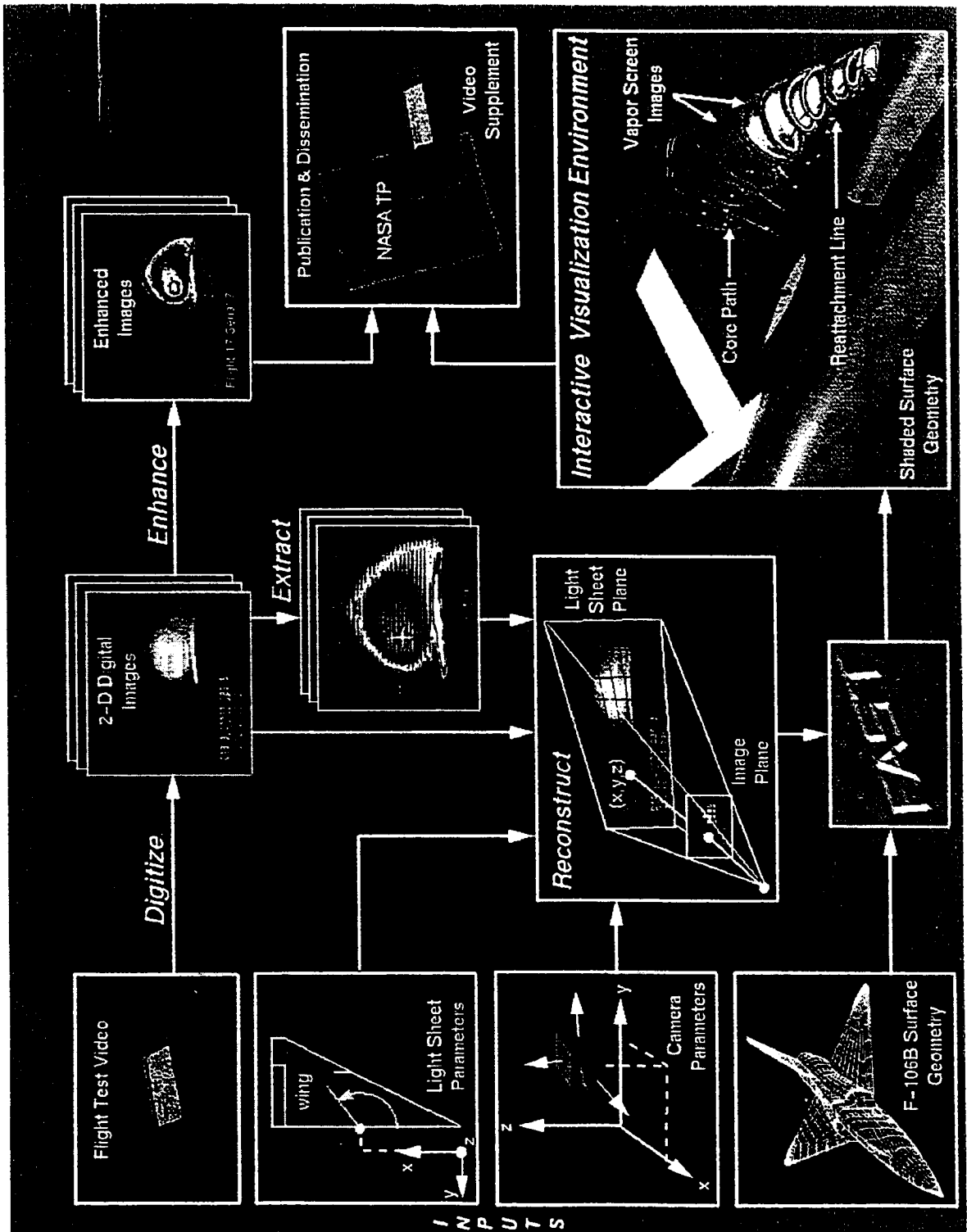
- **F-106B Leading-Edge Flow Visualization Experiment using a Rotating Vapor Screen Technique**
- **Original data source: black & white videotape**
- **DVAL products: a 3-D interactive visual analysis capability
quantitative geometric position information
broadcast quality video**



DVAL OVERVIEW

Sample Application (Cont.)

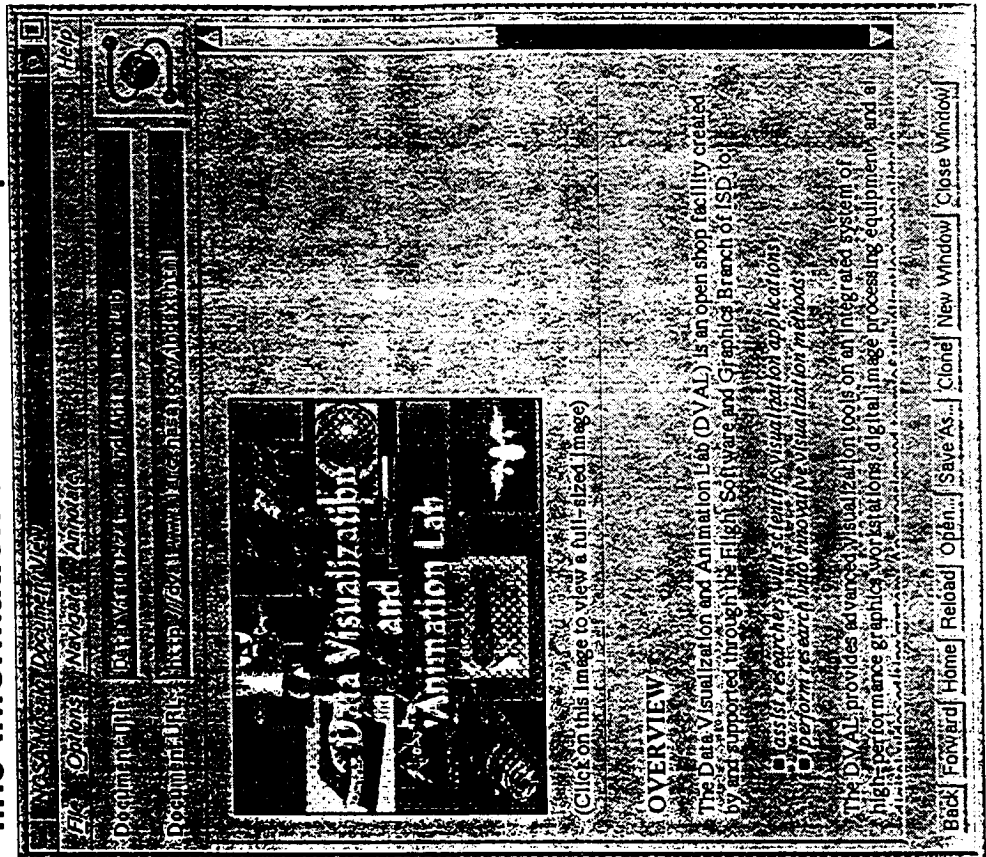




DVAL OVERVIEW

DVAL Mosaic Page

- Up-to-date on-line information about DVAL capabilities



ORIGINAL PAGE IS
OF POOR QUALITY

DVAL OVERVIEW

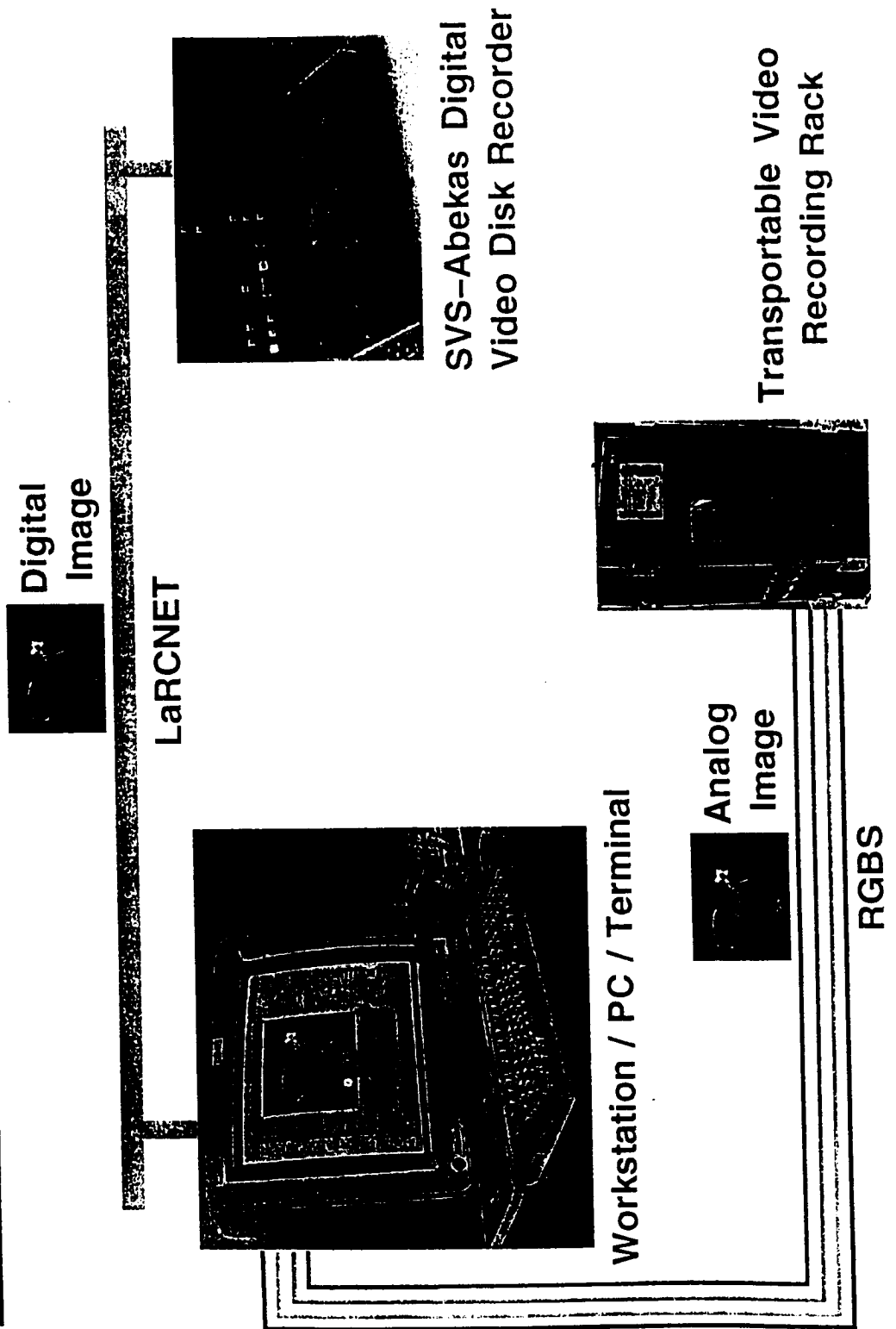
Scientific Visualization System



- Scientific Visualization System (SVS) is a State-of-the-Art Digital Video Editing Suite
- Produce Broadcast-Quality Video Tapes of Computer-Generated Results for :
 - Presentation
 - Analysis
 - Dissemination/Technology Transfer

DVAL OVERVIEW

SVS - Image Acquisition



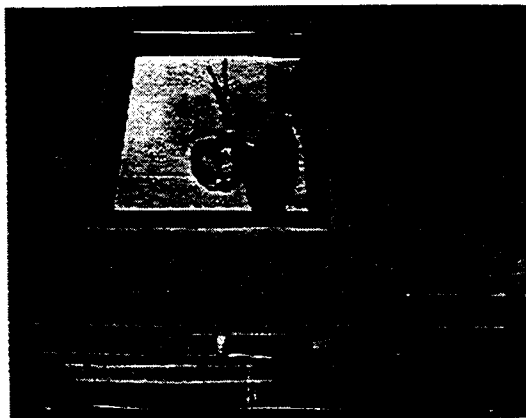
SVS Video Creation

Three Step Process :

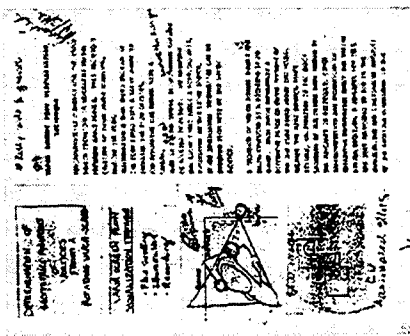
- 1) Pre-Production**
- 2) Production**
- 3) Post-Production**

DVAL OVERVIEW

SVS Pre-Production



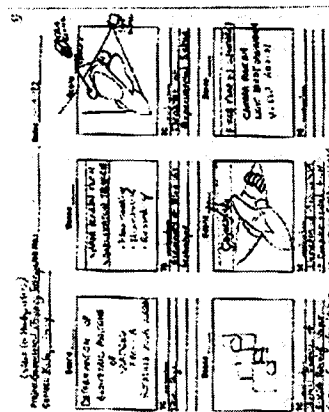
Narration



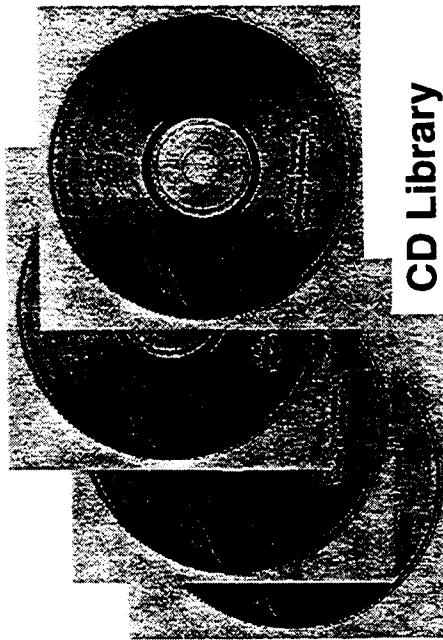
Script Writing and Revision



Music



Storyboard Preparation

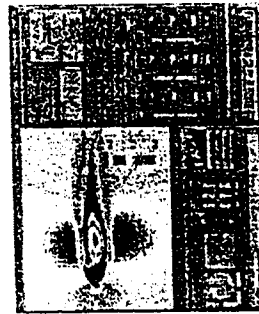


CD Library

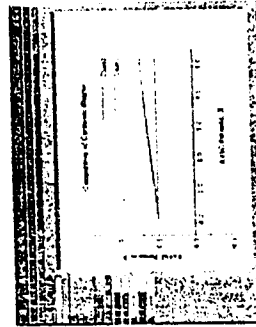
DVAL OVERVIEW

SVS Production

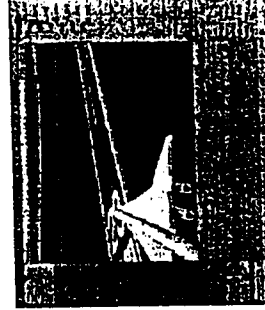
- Computer Programming :
 - Translators
 - Device Drivers
 - Simulators
 - Special Purpose Codes
- Creation of Animations and Images :



FAST



TECPLOT



WAVEFRONT

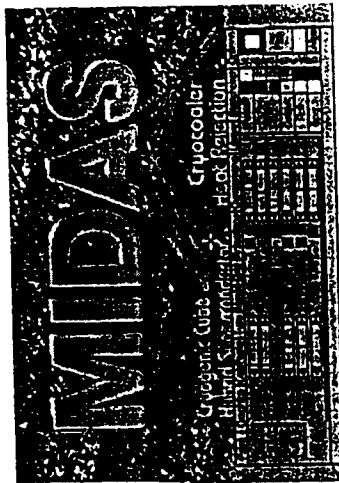
- Transfer of Images to SVS

DVAL OVERVIEW

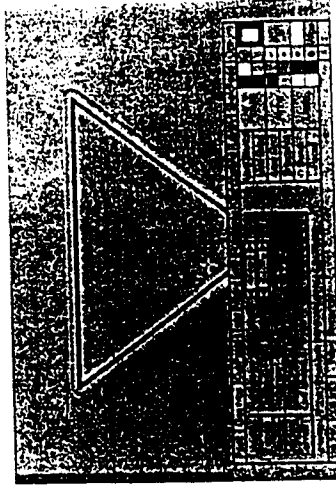
SVS Post-Production

Video Editing :

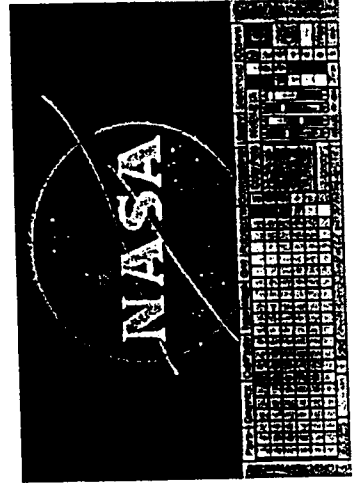
- Cuts
- Fades
- Dissolves
- Wipes



Title Generation



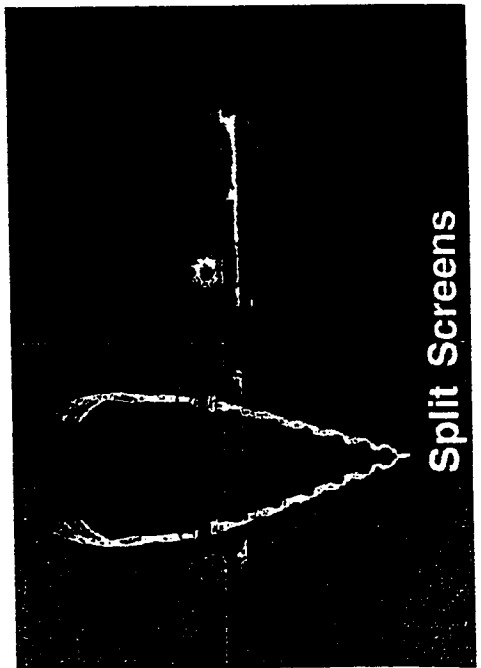
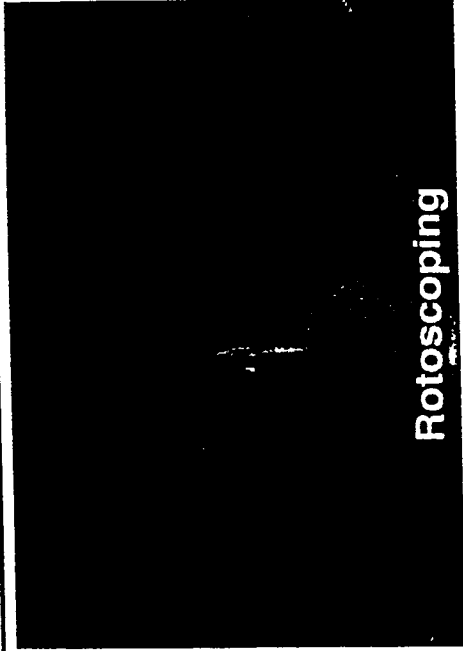
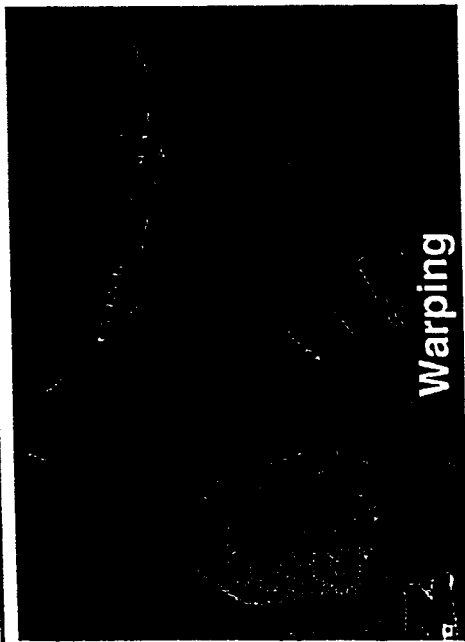
Graphics



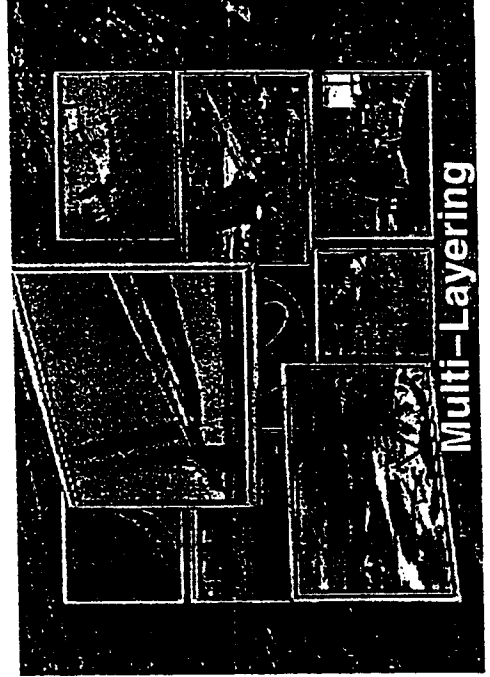
Paint

DVAL OVERVIEW

SVS Post-Production (Cont.)

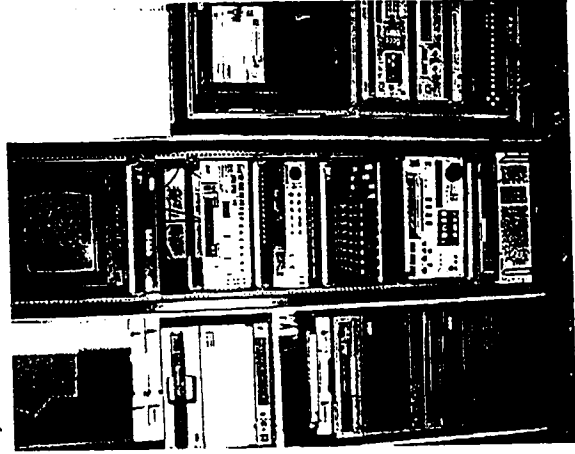
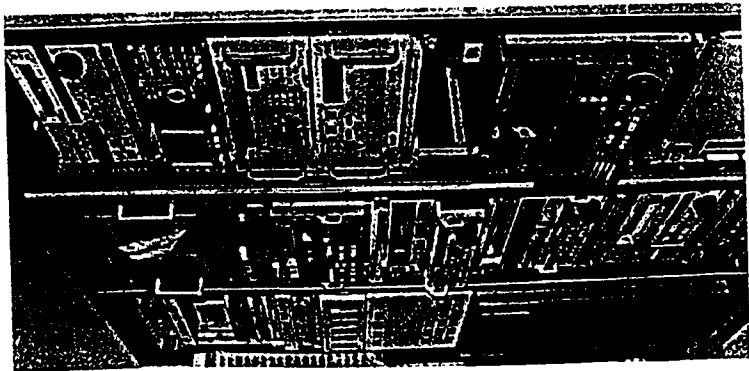


Video Special Effects



SVS Hardware

- DFX Composium Digital Video Editor
- Abekas Digital Video Disk Recorder (2)
- Sony D1 Digital Video Tape Recorder (2)
- Sony Betacam SP Tape Recorder (2)
- Sony Betacam SP Tape Player
- Sony Laser Disk Recorder (2)
- Sony U-matic SP Tape Recorder
- Sony U-matic Tape Recorder
- Panasonic S-VHS Tape Recorder
- JVC VHS Tape Recorder
- Sharp VHS Tape Recorder
- Sony Audio/Video Editor
- Sony Audio Mixer
- Sony CD Player
- JVC DAT Recorder
- Tascam Audio Cassette Recorder
- Acoustic Systems Narration Booth



DVAL OVERVIEW

SVS Misc.

- Creation of MPEG animations from :
 - Computer-generated images
 - Video tape

- Services and material provided by SVS are free of charge

- Contact :

Bill von Offenheim
Scientific Applications Branch
Information Systems Division
x46712
w.h.c.vonoffenheim@larc.nasa.gov

Data Visualization and Animation Lab: Applications

P 23

Kurt Severance, Mike Weisenborn

A wide variety of software tools in DVAL have been successfully used to visualize, analyze, and present computational and experimental data at Langley Research Center. These tools can be roughly categorized according to five primary uses: 2-D image analysis, conventional 3-D visualization, volume visualization, photo-realistic rendering, or special-purpose applications. Software in each of these categories is accessible to LaRC personnel free of charge, and training or consultation can be arranged with the DVAL staff.

Two-dimensional image analysis software is supported on many platforms and provides several fundamental capabilities. The input is generally a 2-D array of bits, bytes, integers, floating-point, or even complex numbers. These arrays can then be represented as color images from which features can be enhanced, extracted, and statistically analyzed. Images can also be represented as contour plots or, by correlating height with a scalar quantity, as 3-D surfaces. Most image analysis tools in use today support two levels of users: the programmer, who intends to incorporate their own algorithms usually through a command-line interface, and the novice end-user who usually prefers to work with a straightforward menu interface. Advanced features include image segmentation and pattern recognition capabilities.

The two primary image analysis packages available in DVAL are PV-WAVE and KB-Vision, both of which have been successfully applied to several LaRC projects. PV-WAVE, a product of Visual Numerics Inc. runs on most UNIX workstations, and multiple licenses are available which can be shared among LaRC researchers. A more advanced product, KB-Vision from Amerinex Artificial Intelligence Inc., employs artificial intelligence techniques to provide state-of-the-art feature extraction and pattern-recognition capabilities. KB-Vision has been used to analyze and reduce images acquired from an in-flight experiment which utilized tufts, and from a spin-tunnel test which utilized retro-reflective targets.

Software which allows interactive visualization of 3-D data was originally specialized for computational fluid dynamics solutions on high-end workstations. Today, tools are available even on modest computing platforms for visualizing dense data from either computational or experimental sources. These conventional tools generally input volume grids, scalar quantities, and vectors in either structured or unstructured format. Arbitrary cross-sectional surfaces through the data can be displayed and colored according to a selected parameter. Advanced features include iso-surfaces (the 3-D extension of a contour), transparency, thresholding, stereo display, and animation.

Three primary scientific visualization packages used at the Center include the Flow Analysis Software Toolkit (F.A.S.T.), Tecplot, and Fieldview. Although these tools were originally intended for CFD research, they have been successfully used to analyze a variety of datasets including those from wind tunnel or in-flight tests, atmospheric simulations, structural analyses, and medical scans. F.A.S.T., a highly interactive environment often used in DVAL to produce animations of 3-D data, is supported on Silicon Graphics (SGI) workstations and is free to all NASA personnel and contractors. Tecplot is free of charge to LaRC personnel and is available on SUN, SGI, DEC, HP, IBM, and PC platforms. Fieldview, by Intelligent Light, Inc., is supported on all major UNIX workstations, and a limited number of licenses are available at LaRC upon request.

Volume visualization techniques offer an alternative to the more traditional visualization tools. Whereas the conventional tools require the user to extract polygonal approximations such as cutting planes and iso-surfaces from their data, volume visualization tools can potentially render an entire volume of data, allowing simultaneous examination of surfaces and internal

structures. This technique is particularly applicable to the analysis of diffuse or "fuzzy" 3-D phenomenon which have no clear boundaries, such as electromagnetic fields. Current volume rendering technology requires the volume to be a rectangular parallelepiped (a box) which is furthermore subdivided into cubic building blocks, called voxels. A value (upto 16 bits) for some measured or calculated property is associated with each voxel. The usefulness of volume visualization has been demonstrated in a number of fields including cell biology, medical imaging, nondestructive testing, molecular modeling, astrophysics, and multi-dimensional mathematics. A high-end volume rendering package called VoxelView by Vital Images, Inc. is supported on SGI and Macintosh platforms and is available for use in DVAL.

When a very high-quality computer-generated image or animation is necessary to describe an otherwise abstract idea or phenomenon, photo-realistic rendering software is required. The Wavefront Advanced Visualizer available in DVAL serves this purpose by providing a menu-driven environment in which such effects as textures, shadows, reflection, refraction, and transparency can be simulated and applied to complex, moving objects. The objects can be modeled within Wavefront or can be imported from other packages such as PLOT3D or IDEAS. This software has proven useful in several areas, primarily in the description of an experimental facilities such as wind tunnels in which interior structures of interest are often inaccessible to conventional cameras. Similarly, many phenomenon which are too small, too large, too abstract, or simply non-accessible have been simulated with this rendering package. Specific applications have included the depiction of a water tunnel experiment, the simulation of shuttle arm flexing due to heavy payloads, the internal structure of multi-layered I-beams, and the simulated take-off of the High Speed Civil Transport. Two copies of the Wavefront Advanced Visualizer are available on DVAL SGI workstations (with 4-CPU's each), and staff is available to produce requested animations or train interested individuals.

A special-purpose application program has been designed in DVAL to support flow visualization experiments which utilize cameras and lightsheets. The software, termed ILLUME (Interactive Lightsheet Locator Utility and Modeling Environment), provides an interactive capability for determining suitable placement of cameras and lightsheets well in advance of the actual experiment and before any instrumentation is configured in a tunnel or on an aircraft. The software allows the user to position cameras and light sheets with respect to the test object or model and see simulated camera views. Adjustments can be made to the camera and light sheet positions and orientations until the desired view is obtained. In addition, roll, pitch, and yaw adjustments can be made to the model to determine whether all desired regions of the model remain visible to the recording camera for a range of test conditions. ILLUME is an OSF/Motif-based program which runs on most SGI workstations, accepts PLOT3D grid and function files, and is freely available to LaRC researchers.

Information about all of the packages mentioned can be obtained through the following e-mail addresses:

PV~WAVE	pvwave-request@hojo.larc.nasa.gov
KB-Vision	isdcs+iphelp@larc.nasa.gov
Tecplot	tecplot@eagle.larc.nasa.gov
Fieldview	j.t.bowen@larc.nasa.gov
F.A.S.T.	isdcs+vizhelp@larc.nasa.gov
VoxelView	isdcs+vizhelp@larc.nasa.gov
Wavefront	isdcs+vizhelp@larc.nasa.gov
ILLUME	isdcs+vizhelp@larc.nasa.gov

**1994 Workshop
The Role Of Computers in LaRC R&D
Graphics and Image Processing Session
June 16, 1994**

DATA VISUALIZATION & ANIMATION LAB (DVAL) APPLICATIONS

**Presented by:
Kurt Severance
Mike Weisenborn**

DVAL APPLICATIONS

Outline

Image Analysis Software

Standard Scientific Visualization Software

Volume Visualization Software

Photo-realistic Rendering Software

Special-purpose Visualization Software

DVAL APPLICATIONS

Image Analysis Software

- inputs arrays of bit, byte, integer, or floating-point data
- color image display, enhancement, feature extraction, and statistics
- line, contour, and surface plots
- programmer or end-user level

PV-WAVE

pvwave-request@hojo



- Visual Numerics, Inc.
- command-line language
- some 3-D analysis capabilities
- most UNIX workstations

KBVision

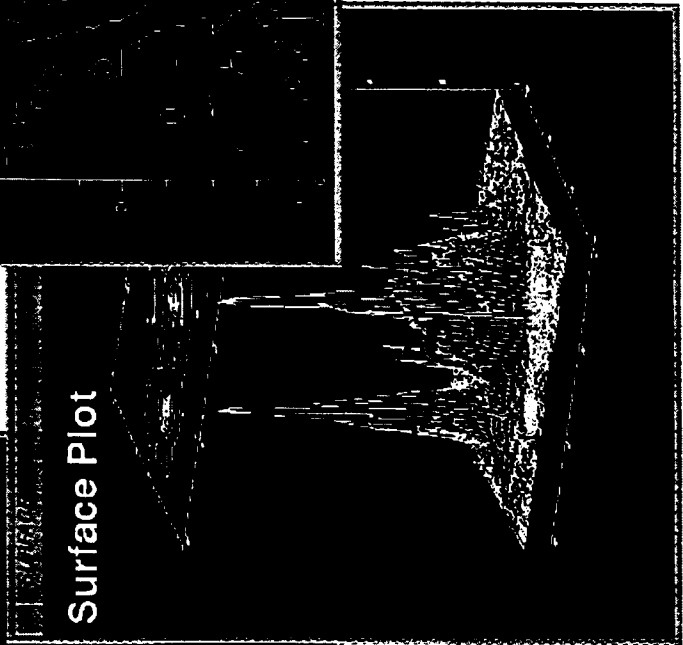
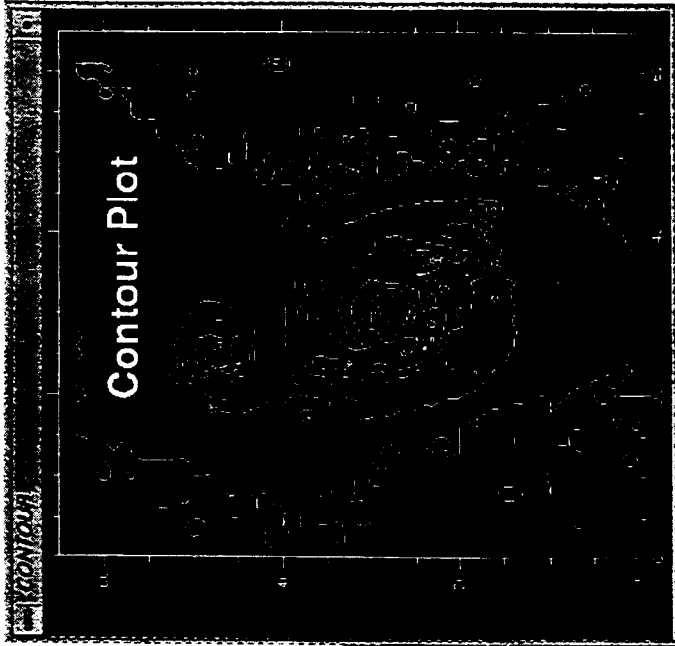
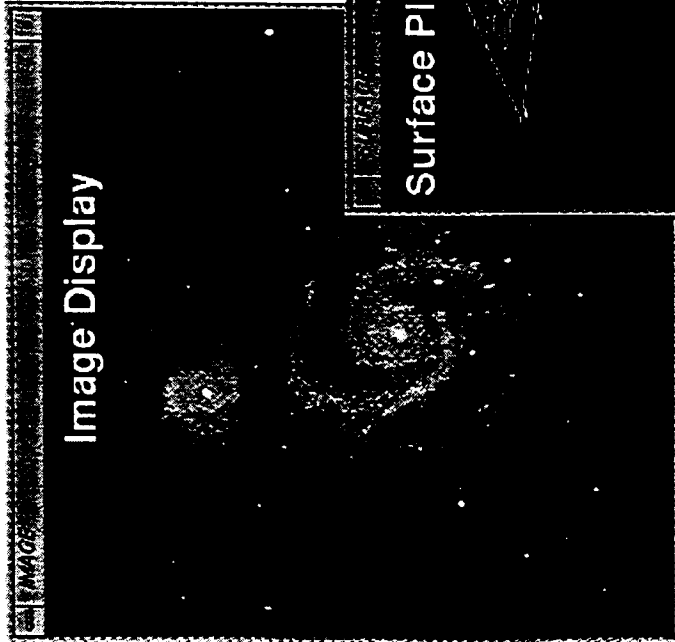
isdcs+iphelp@larc

- Amerinex Artificial Intelligence, Inc.
- employs state-of-the-art A.I. techniques
- well-suited for feature extraction and pattern recognition
- most UNIX workstations

KBVision™

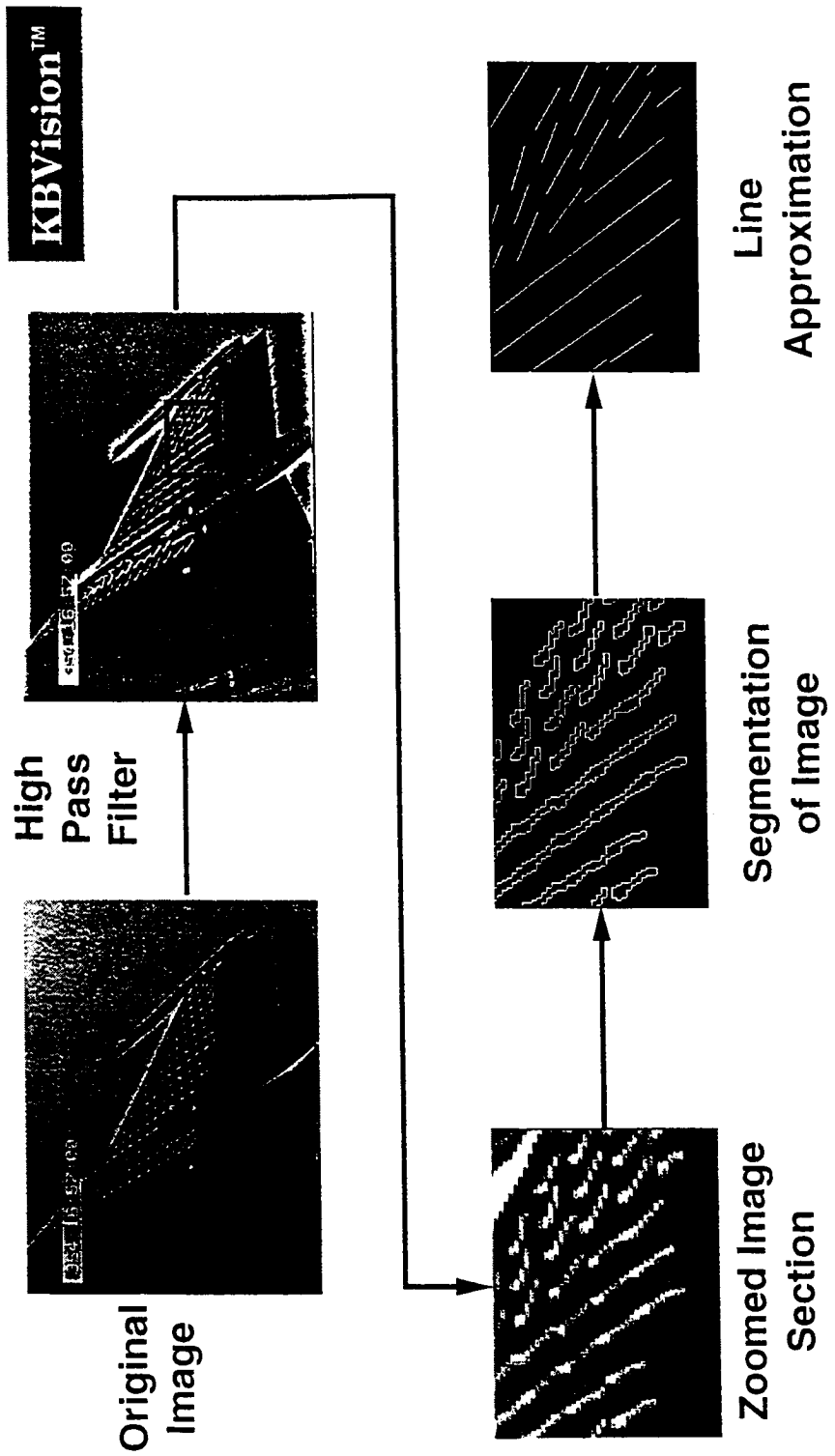
DVAL APPLICATIONS

Image Analysis Example



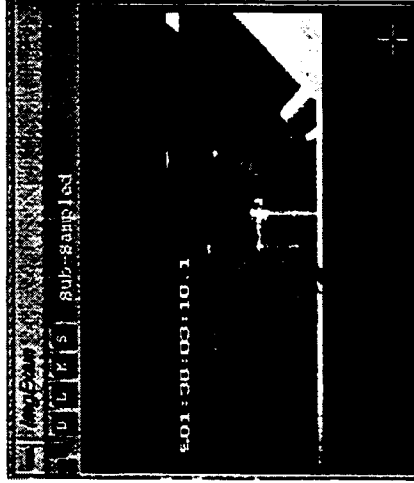
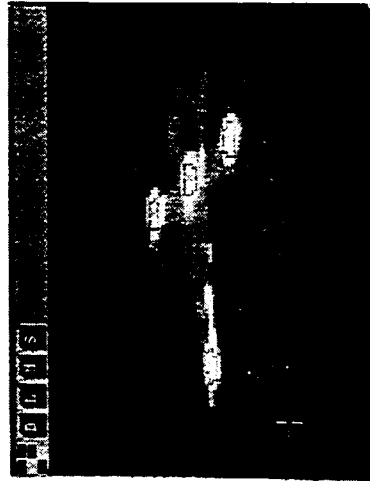
DVAL APPLICATIONS

Feature Extraction Example: Tuft Images

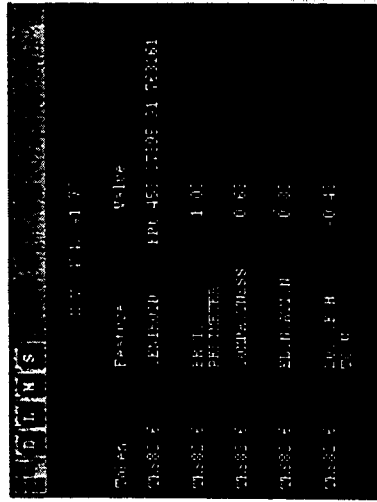
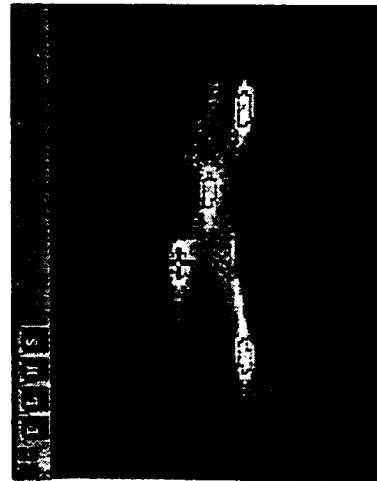
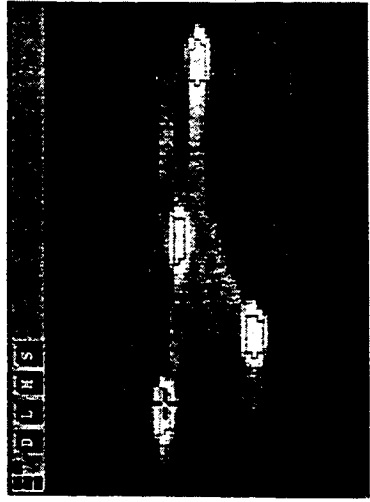


DVAL APPLICATIONS

Feature Extraction Example: Spin Tunnel Images



single video frame



Class	Feature	Value
Class 1	Length	1.00
Class 2	Area	0.80
Class 3	Perimeter	0.80
Class 4	Volume	0.80
Class 5	Surface	0.80

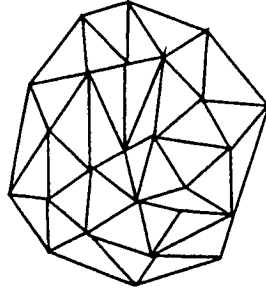
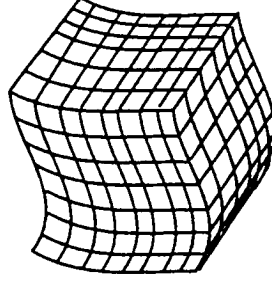


DVAL APPLICATIONS

Standard Scientific Visualization Software

HISTORY:

- evolved from software originally designed to analyze CFD results on specialized workstations
- now available on a variety of platforms and useful in analyzing practically any experimental or computational data



GENERAL CAPABILITIES:

- 3-D structured or unstructured grids:
- highly interactive environment
- arbitrary cross-sectional surfaces through the data
- contoured or colored surfaces according to scalar quantities
- iso-surfaces
- vector fields
- particle traces

Standard Scientific Visualization Software

WIDELY-USED PACKAGES:

Flow Analysis Software Toolkit (F.A.S.T)

- Ames Research Center / Sterling Software
- only SGI workstations
- free to NASA

isdcs+vizhelp@larc

Fieldview

- Intelligent Light, Inc.
- all major UNIX workstations
- limited number of licenses available

j.t.bowen@larc

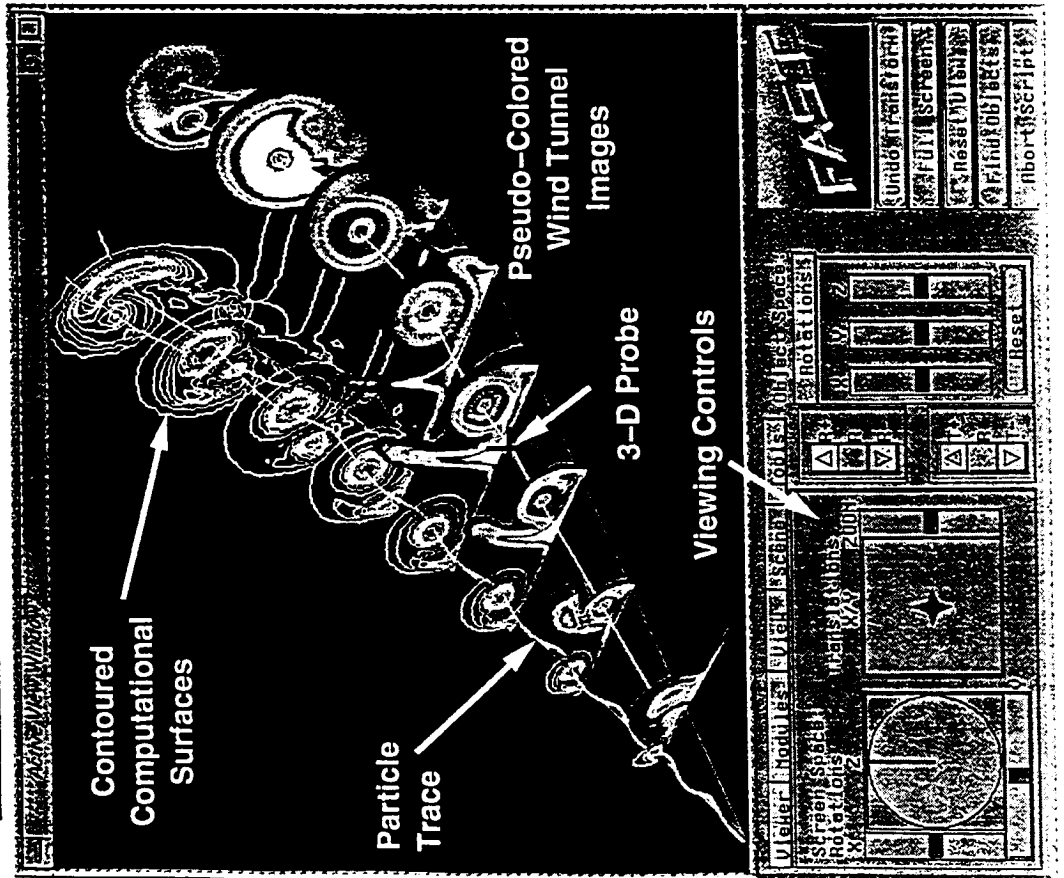
Tecplot

- Amtec Engineering, Inc.
- SUN, SGI, DEC, HP, IBM, PC
- available at LaRC upon request

tecplot@eagle

DVAL APPLICATIONS

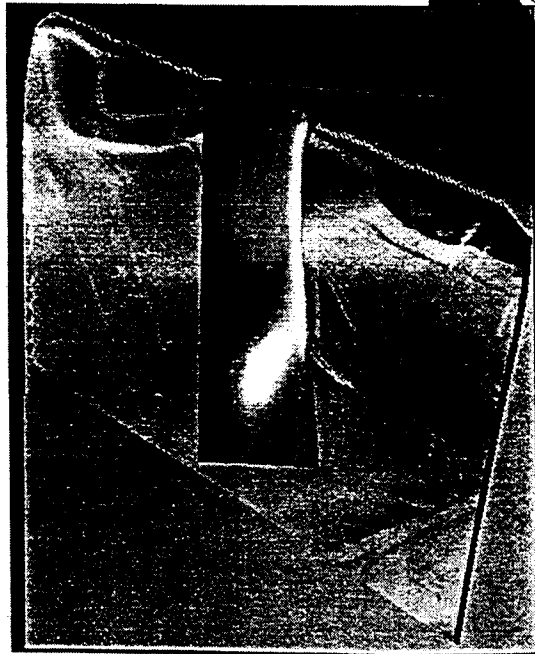
Flow Analysis Software Toolkit



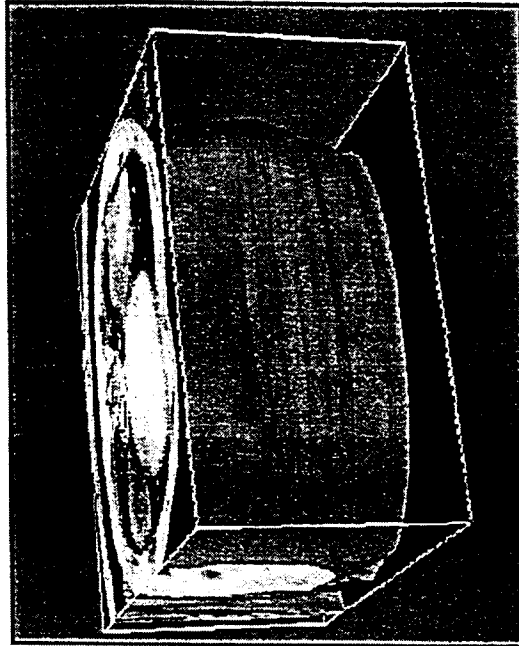
DVAL APPLICATIONS

Other Applications

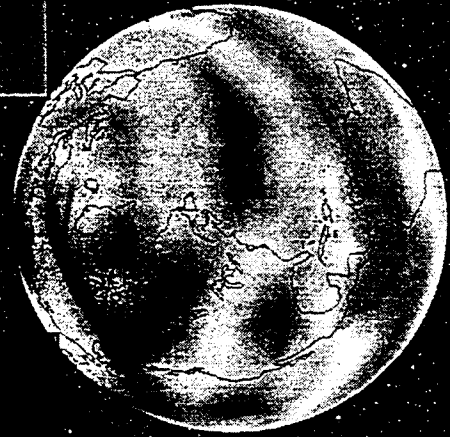
Flight Data



Medical Data



Atmospheric Data



DVAL APPLICATIONS

Volume Visualization Software

FEATURES:

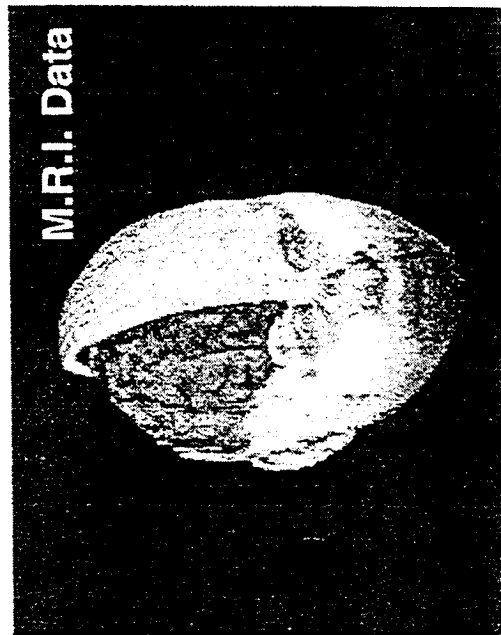
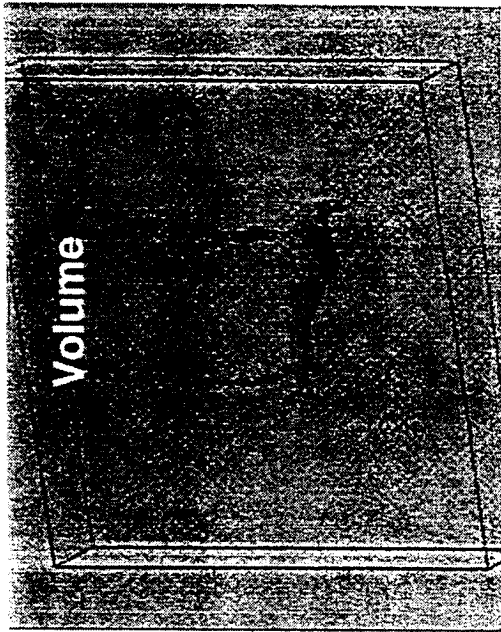
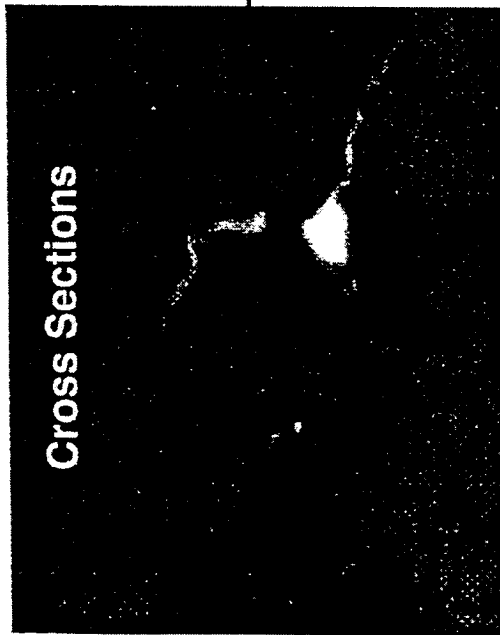
- combines the concepts of image processing and computer graphics to visualize an entire volume of scalar data
- requires the volume to be subdivided into regularly-spaced cubes
- well-suited for "fuzzy" or diffuse distributions of 3-D data

ADVANTAGES OVER CONVENTIONAL (POLYGONAL) TECHNIQUES:

- unnecessary to create a polygonal approximation
- allows phenomena without clear boundaries (i.e., energy fields) to be represented accurately
- internal structures easily examined using transparency

AVAILABLE IN DVAL: VoxeIView, by Vital Images, Inc.

VoxelView: Examples

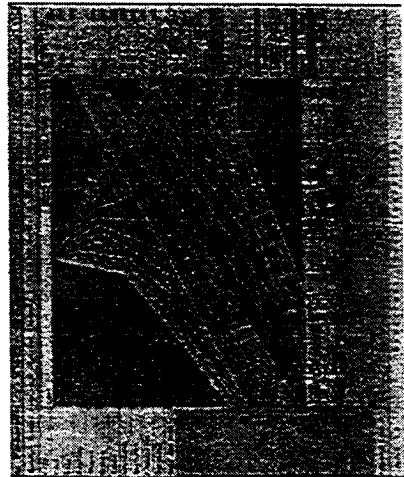


DVAL APPLICATIONS

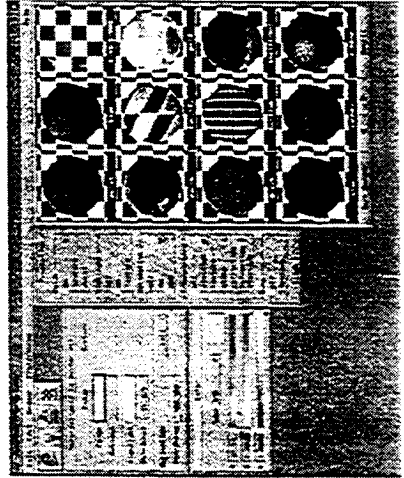
Wavefront Advanced Visualizer

- Photorealistic Computer Animation
- Used for Still Images and Animation Sequences

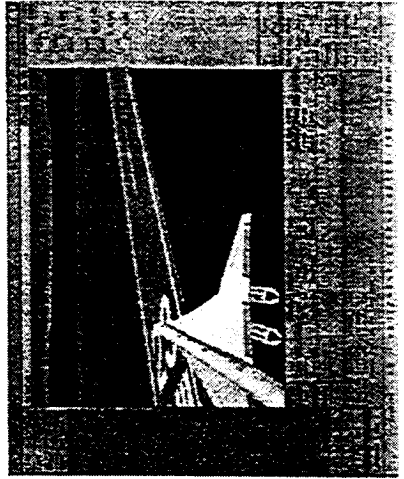
Object Modeling



Material Editing



Motion Definition



ILLUME Controls

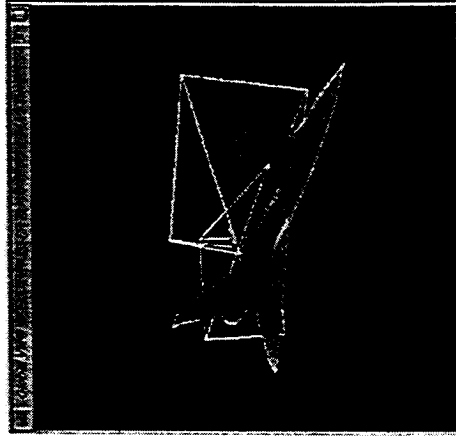
- Camera position and orientation
- Up to 6 Cameras
- Lightsheet position, orientation, and type (rotating or translating)
- Up to 6 Lightsheets
- Model translation, roll, pitch, and yaw
- Uses Plot3D gridfiles for geometry
- Allows function mapping onto grids

DVAL APPLICATIONS

ILLUME Views

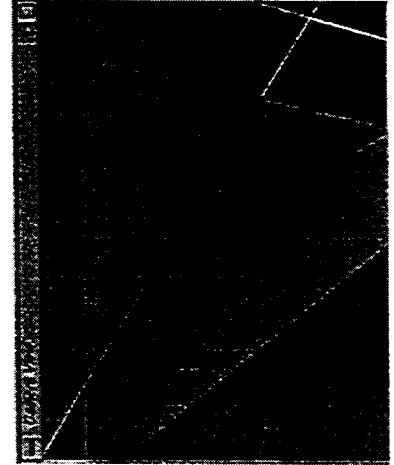
- **Global View:**

Shows overall setup
of experiment



- **Camera View:**

Shows accurate view
through camera



DVAL APPLICATIONS

ILLUME Availability

- **Runs on most Silicon Graphics workstations**
- **Available on DVAL machines**
- **Freely available to LaRC researchers**
- **DVAL training and support**

SESSION 8 System Design and Integration

Chaired by

Jerry H. Tucker

- 8.1 The Design Manager's Aid for Intelligent Decomposition DeMAID - Jim Rogers
- 8.2 RDD-100 and the Systems Engineering Process - Robert Averill
- 8.3 Computer Tools for Systems Engineering at LaRC - J. Milam Walters
- 8.4 A Distributed Computing Environment for Multidisciplinary Design FIDO - Robert Weston
- 8.5 An Overview of the Computer Aided Engineering and Design for Electronics Laboratory CAEDE - Shelley Stover
- 8.6 The Software Engineering and/or Ada Lab (SEAL) - Robert Kudlinski

Scientific Applications

- • Display Experimental Setup
- Demonstrate Unobservable Phenomena
- Show Internal Structure of Objects
- Visually Simulate Future Technology

WATER TUNNEL SETUP

- Photographs can not fully demonstrate the experimental setup.
- Computer animation can dissolve certain features to clarify setup.



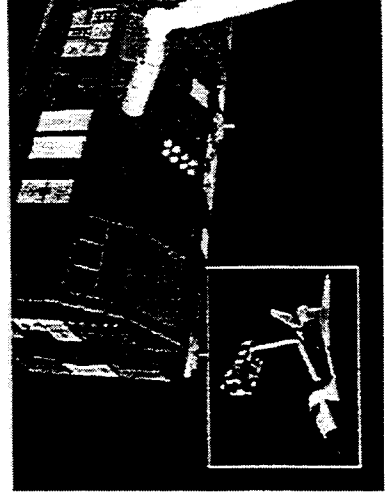
DVAL APPLICATIONS

Scientific Applications

- Display Experimental Setup
- • Demonstrate Unobservable Phenomena
- Show Internal Structure of Objects
- Visually Simulate Future Technology

SHUTTLE ARM CONTROL

- Researchers were investigating methods to control flex in arm.
- Computer animation allowed exaggeration and demonstration of flex.



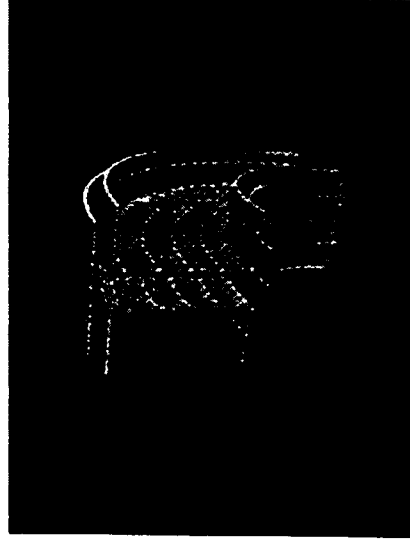
DVAL APPLICATIONS

Scientific Applications

- Display Experimental Setup
- Demonstrate Unobservable Phenomena
- Show Internal Structure of Objects
- Visually Simulate Future Technology

I-BEAM STRUCTURE

- I-beams were constructed with multiple layers.
- Computer animation allowed demonstration of internal structure.



DVAL APPLICATIONS

Scientific Applications

- Display Experimental Setup
- Demonstrate Unobservable Phenomena
- Show Internal Structure of Objects
- • Visually Simulate Future Technology

HIGH SPEED CIVIL TRANSPORT

- Aircraft is still in design phase.
- Computer animation provided a preliminary view of the aircraft.



DVAL APPLICATIONS

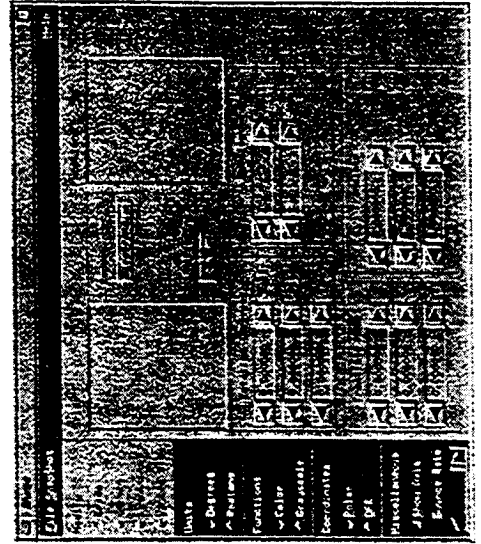
Wavefront in DVAL

- Two copies of the Advanced Visualizer
- Running on 4-CPU Silicon Graphics Workstations
- Assistance is available for learning Wavefront
- DVAL staff with expertise in using Wavefront

DVAL APPLICATIONS

ILLUME

- Interactive Lightsheet Locator Utility and Modeling Environment
- Designed to ease setup of experiments involving cameras and lightsheets.
- Developed in DVAL – local expertise
- Graphical User Interface (MOTIF)



The Design Manager's Aid for Intelligent Decomposition DeMAID

P. 34

James L. Rogers
Ext. 42810

Many engineering systems are large and multi-disciplinary. Before the design of new complex systems such as large space platforms can begin, the possible interactions among subsystems and their parts must be determined. Once this is completed the proposed system can be decomposed to identify its hierarchical structure. DeMAID (A Design Manager's Aid for Intelligent Decomposition) is a knowledge-based system for ordering the sequence of modules and identifying a possible multilevel structure for the design problem. DeMAID displays the modules in an $N \times N$ matrix format (called a design structure matrix) where a module is any process that requires input and generates an output. (Modules which generate an output but do not require an input, such as an initialization process, are also acceptable.) Although DeMAID requires an investment of time to generate and refine the list of modules for input, it could save a considerable amount of money and time in the total design process, particularly in new design problems where the ordering of the modules has not been defined.

The decomposition of a complex design system into subsystems requires the judgment of the design manager. DeMAID reorders and groups the modules based on the interactions among the modules, helping the design manager make decomposition decisions early in the design cycle. These interactions can be deleted interactively. The modules are grouped into circuits (the subsystems) and displayed in a design structure matrix format. Feedbacks, which indicate an iterative process, are minimized and only occur within a subsystem. Since there are no feedback links among the circuits, the circuits can be displayed in a multilevel format. Thus, a large amount of information is reduced to one or two displays which are stored for later retrieval and modification. The design manager and leaders of the design teams then have a visual display of the design problem and the intricate interactions among the different modules.

The design manager could save a substantial amount of time if circuits on the same level of the multilevel structure are executed in parallel. DeMAID estimates the time savings based on the number of available processors. In addition to decomposing the system into subsystems, DeMAID examines the dependencies of a problem with design and behavior variables and creates a dependency matrix. This matrix shows the relationship among the independent design variables and the dependent constraint and objective functions. DeMAID is based on knowledge base techniques to provide flexibility and ease in adding new capabilities. Although DeMAID was originally written for design problems, it has proven to be very general in solving any problem which contains modules (processes) which take an input and generate an output.

The user begins the design of a system by determining the level of modules which need to be ordered. The level is the "granularity" of the problem. The design manager may wish to examine disciplines (a coarse model), analysis programs, or the data level (a fine model). Once the system is divided into these modules, the user determines each module's input and output, creating a data file for the main program. DeMAID is executed through a system of menus. The user has the choice to plan, schedule, display the design structure matrix, display the multilevel organization, examine parallelism, examine the dependency matrix, or trace the effects of a change in the design process. The main program calls a subroutine which reads a rule file and a data file, asserts facts into the knowledge base, and executes the CLIPS inference engine. All DeMAID code is in C for portability.

There are several new capabilities planned for DeMAID. Currently, interactions either exist or not with no quantification as to their strength. Sensitivity analysis is to be used to determine whether or not the interface between two modules is strong or weak. Weak interfaces may be deleted or suspended, thereby reducing iteration times. A second capability will allow the user to breakdown the output of a module into several important pieces so individual pieces can be traced as opposed to the entire output. Currently, DeMAID orders modules within a circuit based on minimizing the number of feedbacks. Since several different orderings may produce the same number of feedbacks, a genetic algorithm is being considered to find the optimal ordering based on a user-defined cost function. Finally, a graphical user interface is being added to make DeMAID more user-friendly.

THE DESIGN MANAGER'S AID FOR INTELLIGENT DECOMPOSITION (DeMAID)

James L. Rogers

**Presented at the Workshop on
The Role of Computers in LaRC R&D
June 15-16, 1994**

OUTLINE

The problem

Design structure matrix

Overview of DeMAID

Current capabilities

Design trade-offs

New capabilities

Summary

THE PROBLEM

Designing a new and complex, multidisciplinary system such as the conceptual design of a high speed aircraft requires:

- (1) Determining the interactions among the computational and data modules
- (2) Reordering the sequence of the modules to minimize feedback
- (3) Dividing the design work among the design teams

Very few tools available to aid the design manager in making early design decisions

QUOTES FROM DESIGNING THE DESIGN PROCESS BY DR. DANIEL E. WHITNEY

"Design is 100 people in a room arguing."

"Right now product designers have all the fancy computer tools. This gives them an unfair advantage when negotiating with manufacturing people."

"We don't have a problem. It's those dummies in manufacturing."

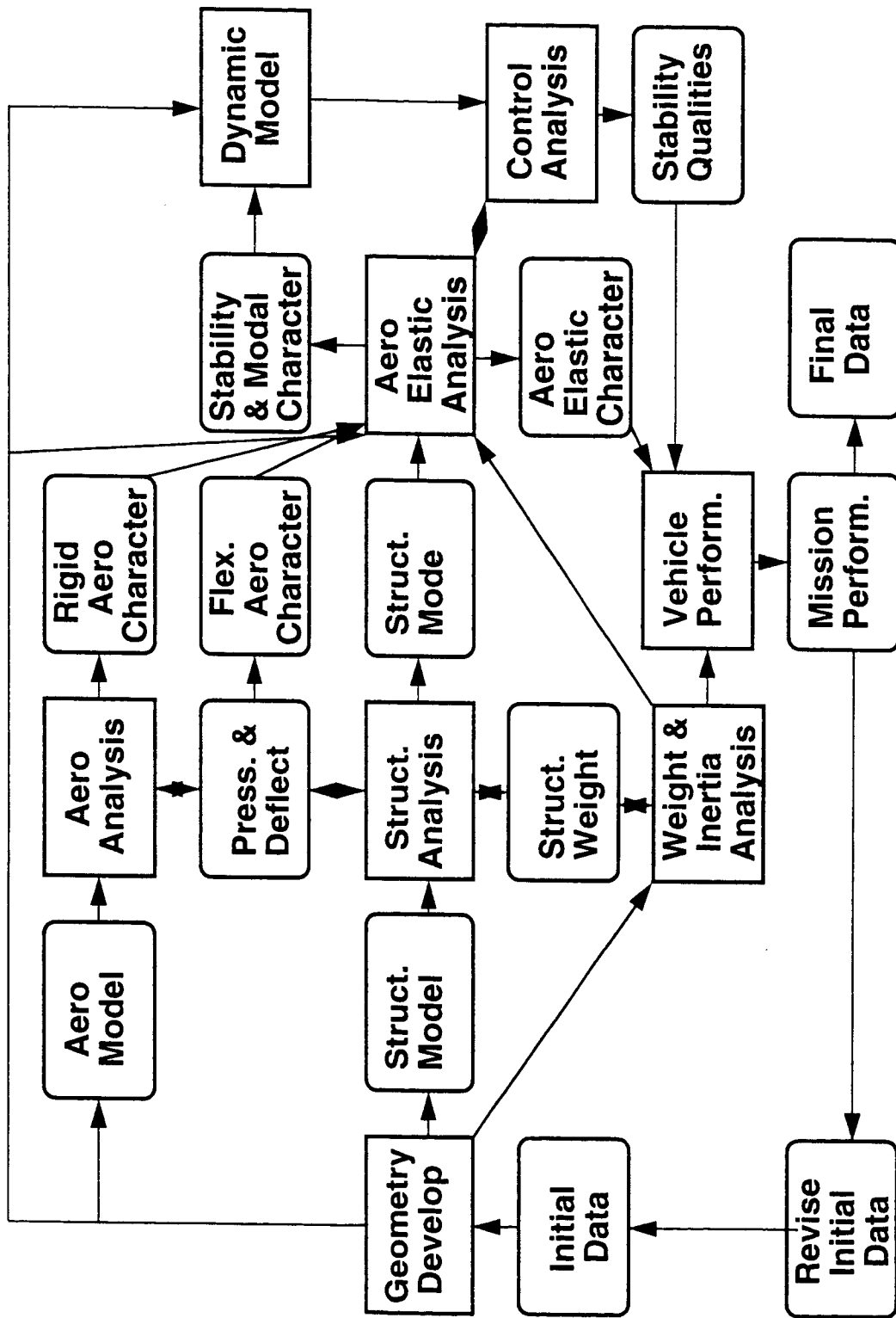
"Engineering proposes designs. Manufacturing counters with problems. There is a lot of conflict, but it is creative."

"Nissan designs all of the parts of an assembly at once while (US auto maker) designs them all in series. How do they ever finish?"

"Word came down from headquarters that we were to eliminate screws. So we redesigned it with all snap fits. Then shipping told us it had to pass a drop test. So we dropped it and it fell apart."

"Just because something can be made doesn't mean it can be manufactured"

PROCESS CHART OF ANALYSES



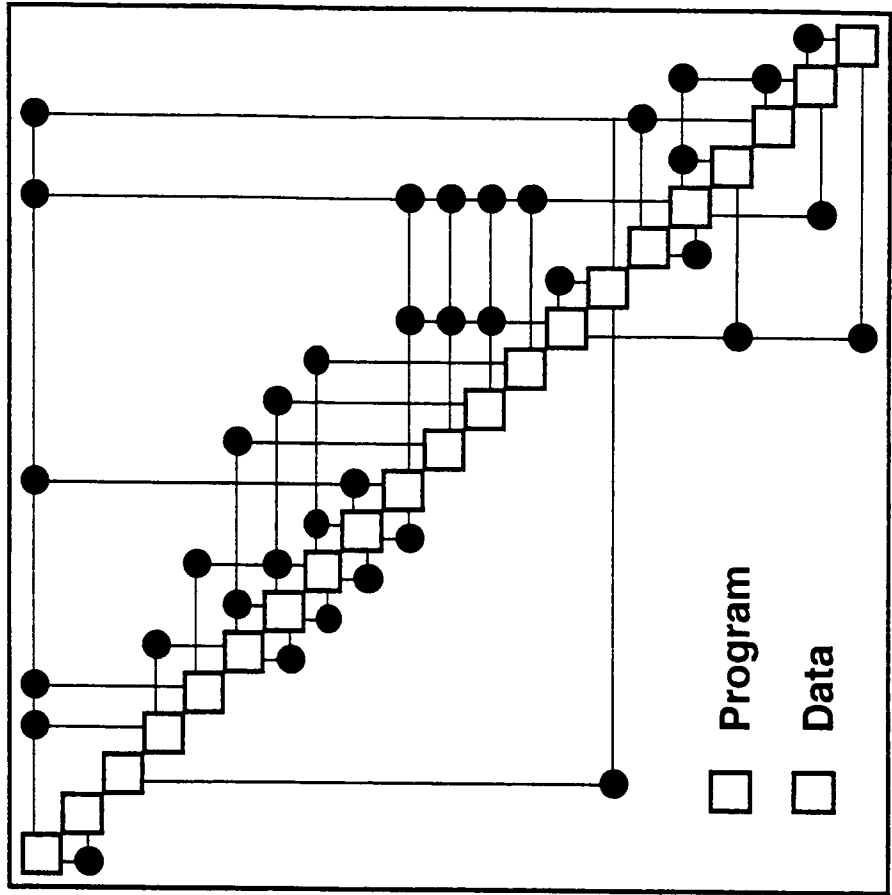
**QUOTES FROM
"MODEL-BASED APPROACHES TO
MANAGING CONCURRENT
ENGINEERING"
BY STEVEN D. EPPINGER**

"Concurrent engineering does not simplify the design process: rather it adds a tremendous amount of inter-task coupling which makes the overall job considerably more difficult."

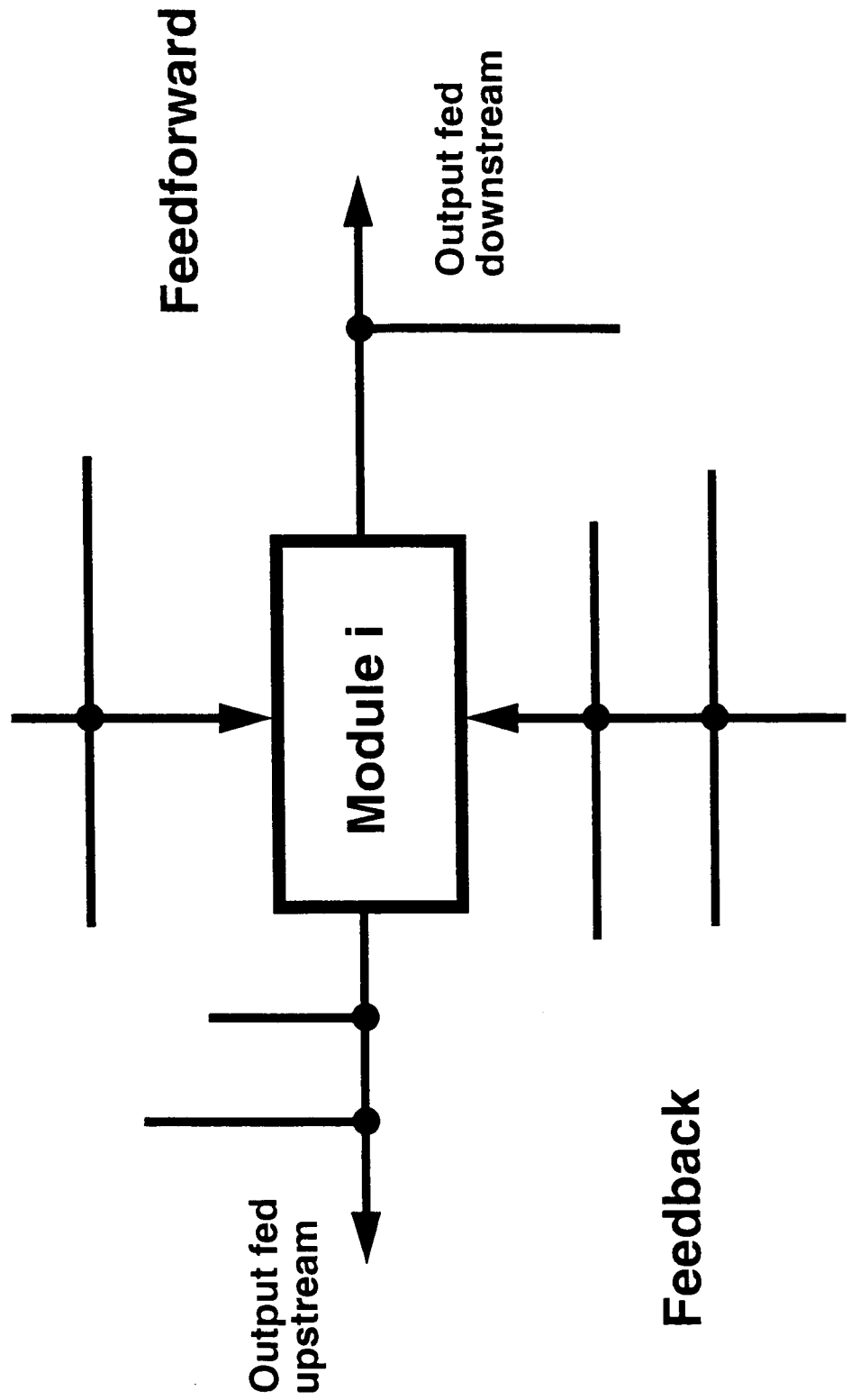
"Concurrent engineering succeeds in reducing overall design time only when adding earlier iteration eliminates later iteration which would have taken even longer."

"The design structure matrix is a tool which allows groups to visualize the relationships among their various activities and reach consensus regarding which feedbacks are to be allowed. However, this tool does not actually show how to alter the process, it merely provides a framework for analyzing alternatives."

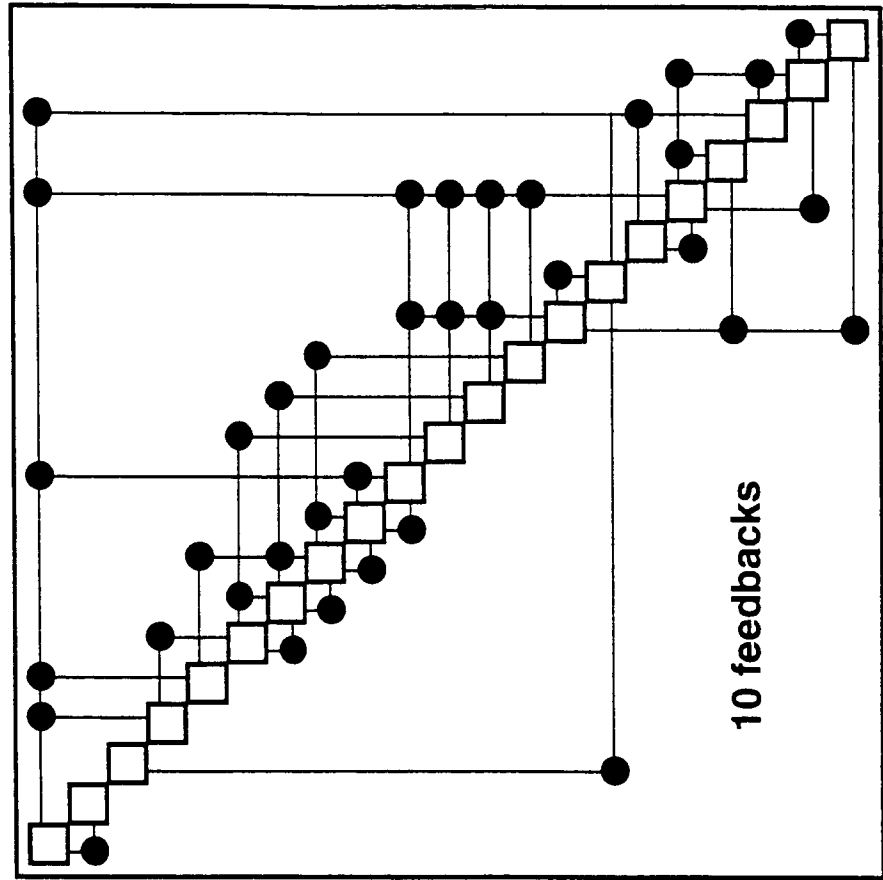
DESIGN STRUCTURE MATRIX (UNORDERED)



MODULE IN DESIGN STRUCTURE MATRIX



DESIGN STRUCTURE MATRIX (UNORDERED)



DESIGN STRUCTURE MATRIX CIRCUIT

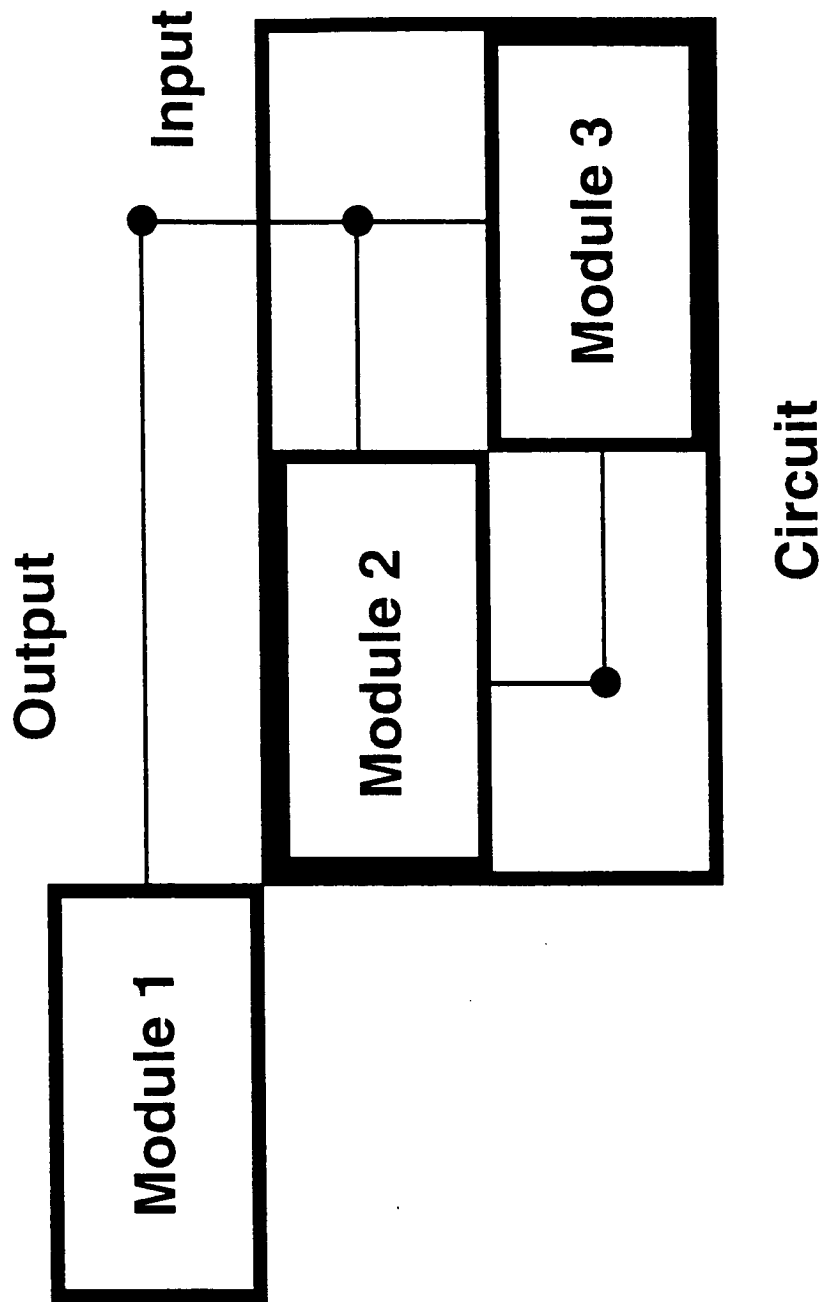
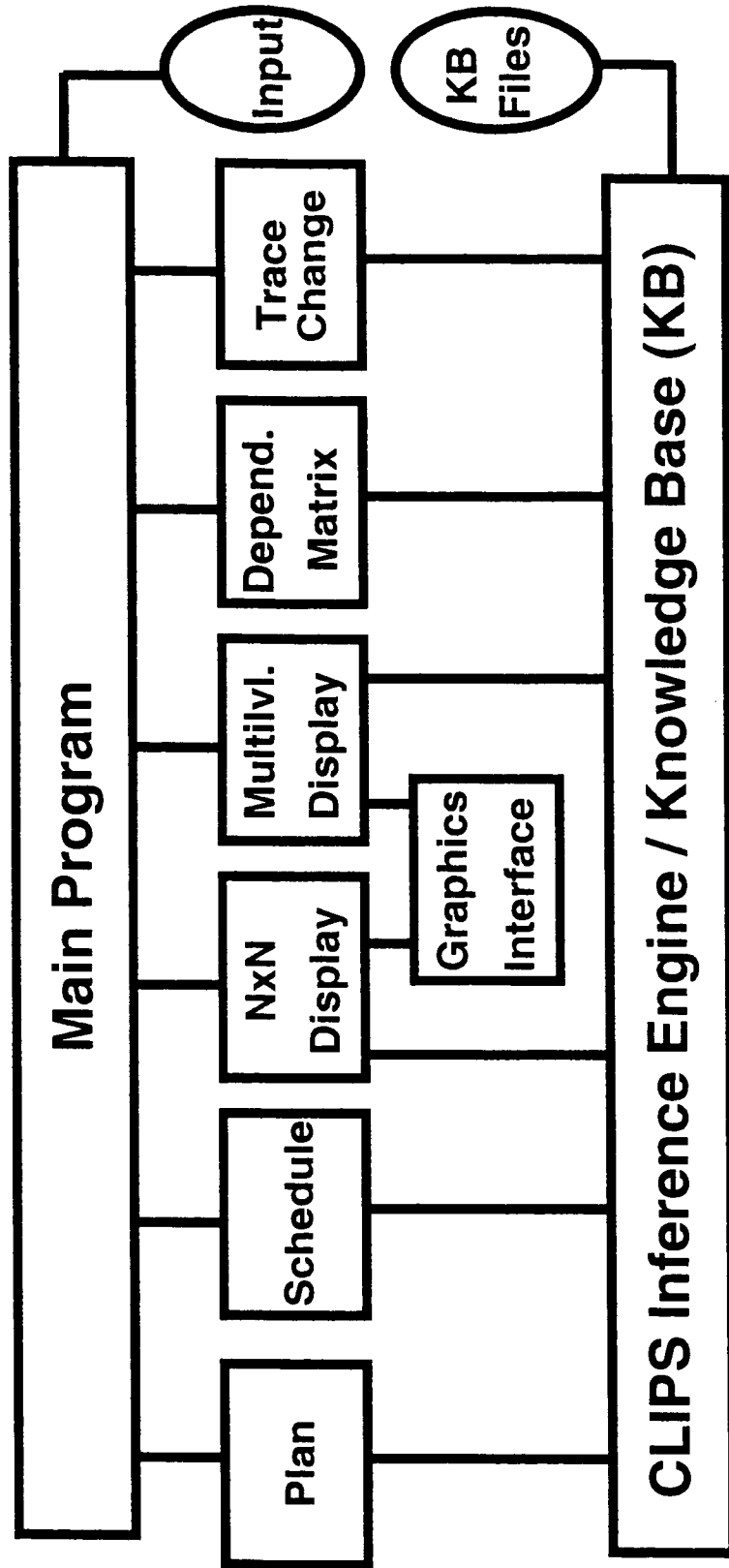


DIAGRAM OF DeMAID

(Design Manager's Aid for Intelligent Decomposition)



DEFINING A MODULE FOR DeMAID

Output = Process (Input1, Input2, . . . , Inputn)

Example:

Output - Performance

Process - ASP (a computer program)

Input - Mission requirements, surface geometry,
flap definitions, aerodynamic forces, weight

Time - 34 minutes

Type - 2

Status - uk (unknown)

Sample input:

module 1 ASP 2 34 PERF uk MR SG FD AF WT

PLANNING FUNCTION

**Remove modules not contributing to the solution
(modules with output not required as input
by another module)**

**Add modules not in original list to satisfy
input requirements**

SCHEDULING FUNCTION

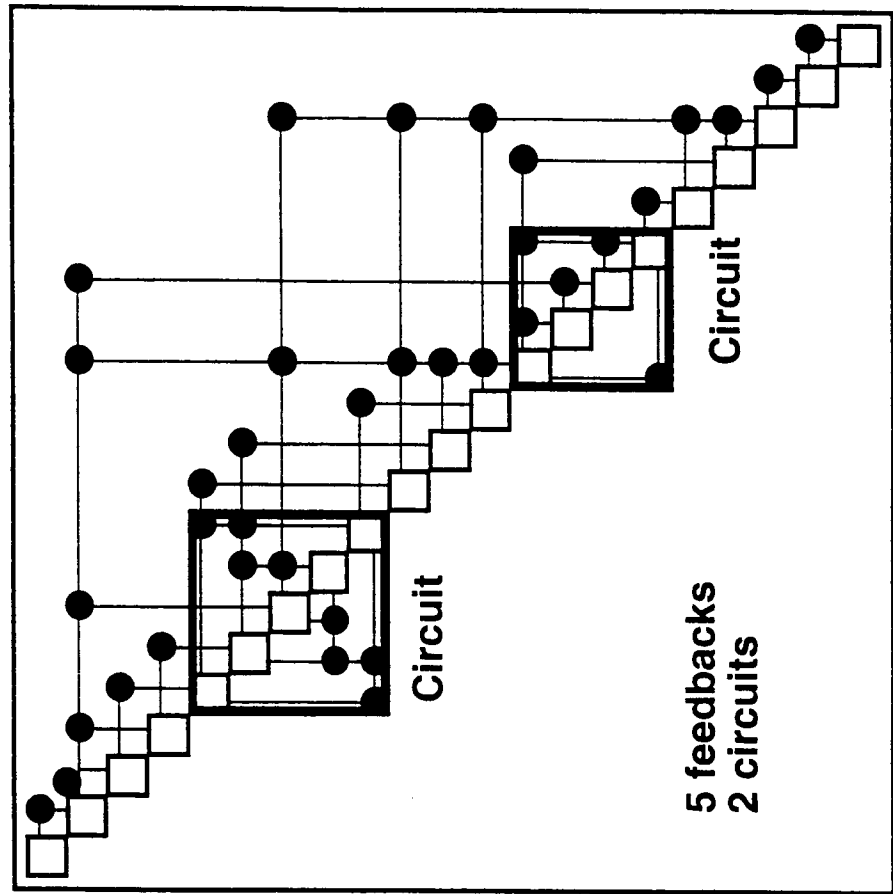
Order modules based on input / output requirements

Minimize feedbacks

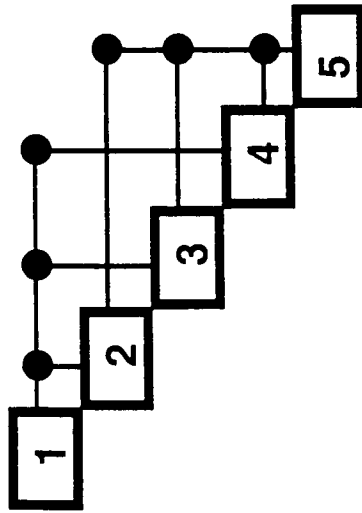
Group modules into circuits

**Place modules into design structure matrix
format for display**

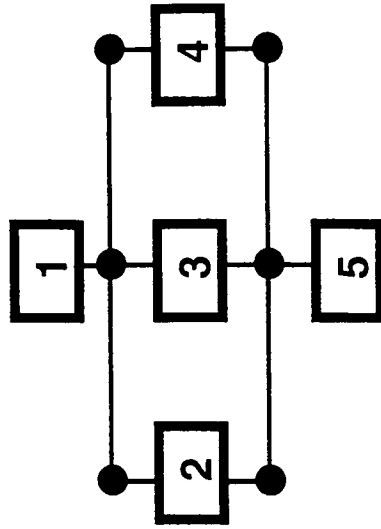
DESIGN STRUCTURE MATRIX (ORDERED)



PROBLEM DECOMPOSITION



**Design Structure Matrix
of Circuits**



**Hierarchical Display
with Parallelization**

DEPENDENCY MATRIX

Design Variables

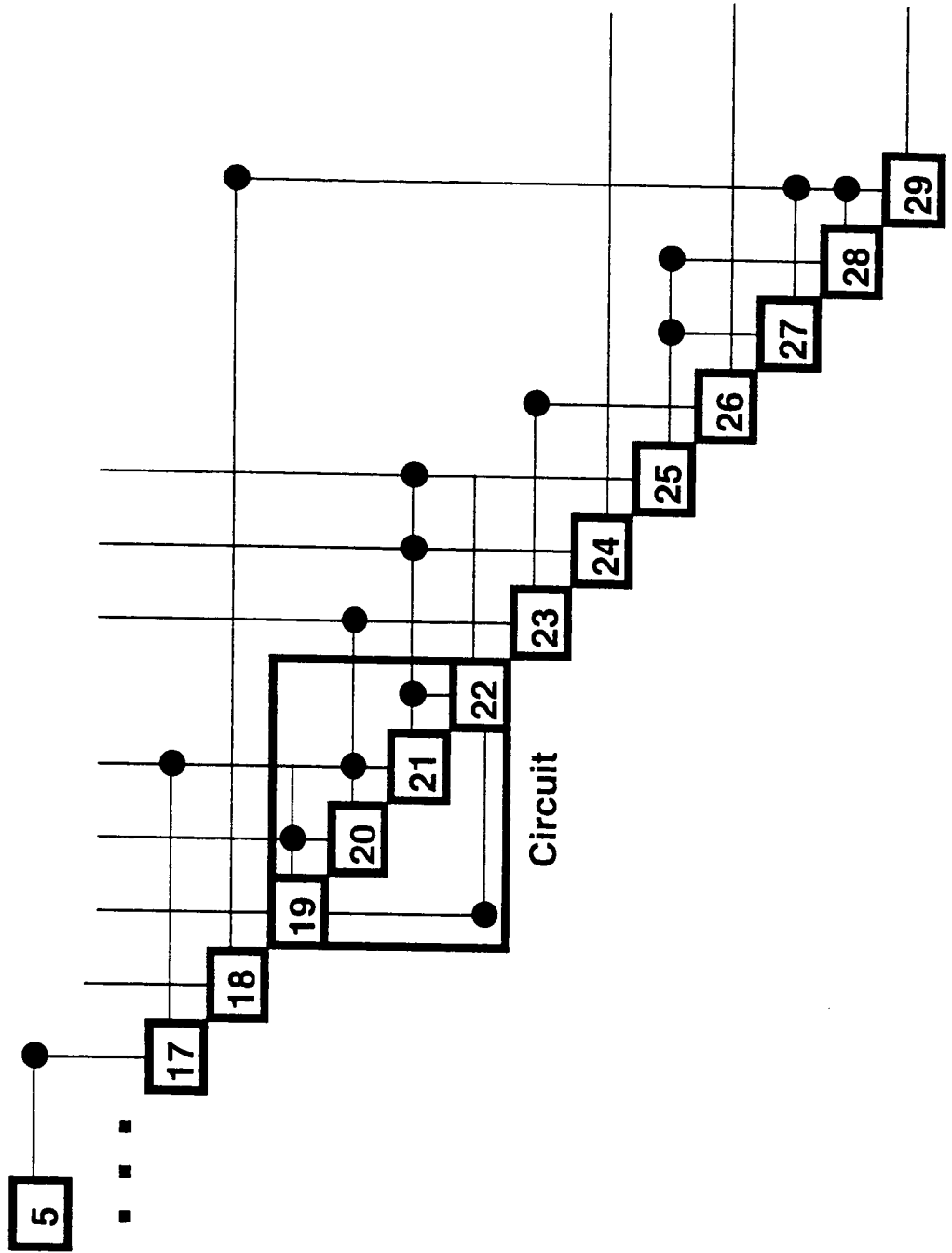
Module	2	3	4	5	10	11	13	14	15
C	X	X	X	X					
O	X	X	X	X					
n									
s									
t		X	X	X					
r									
a		X	X	X	X	X			
i									
n									
t	X						X	X	X
s	X						X	X	X

MAKING DESIGN CHANGES

Just because a change is made to some data
in the design process,

this does not mean that all the modules in the
system must be re-executed

DESIGN STRUCTURE MATRIX

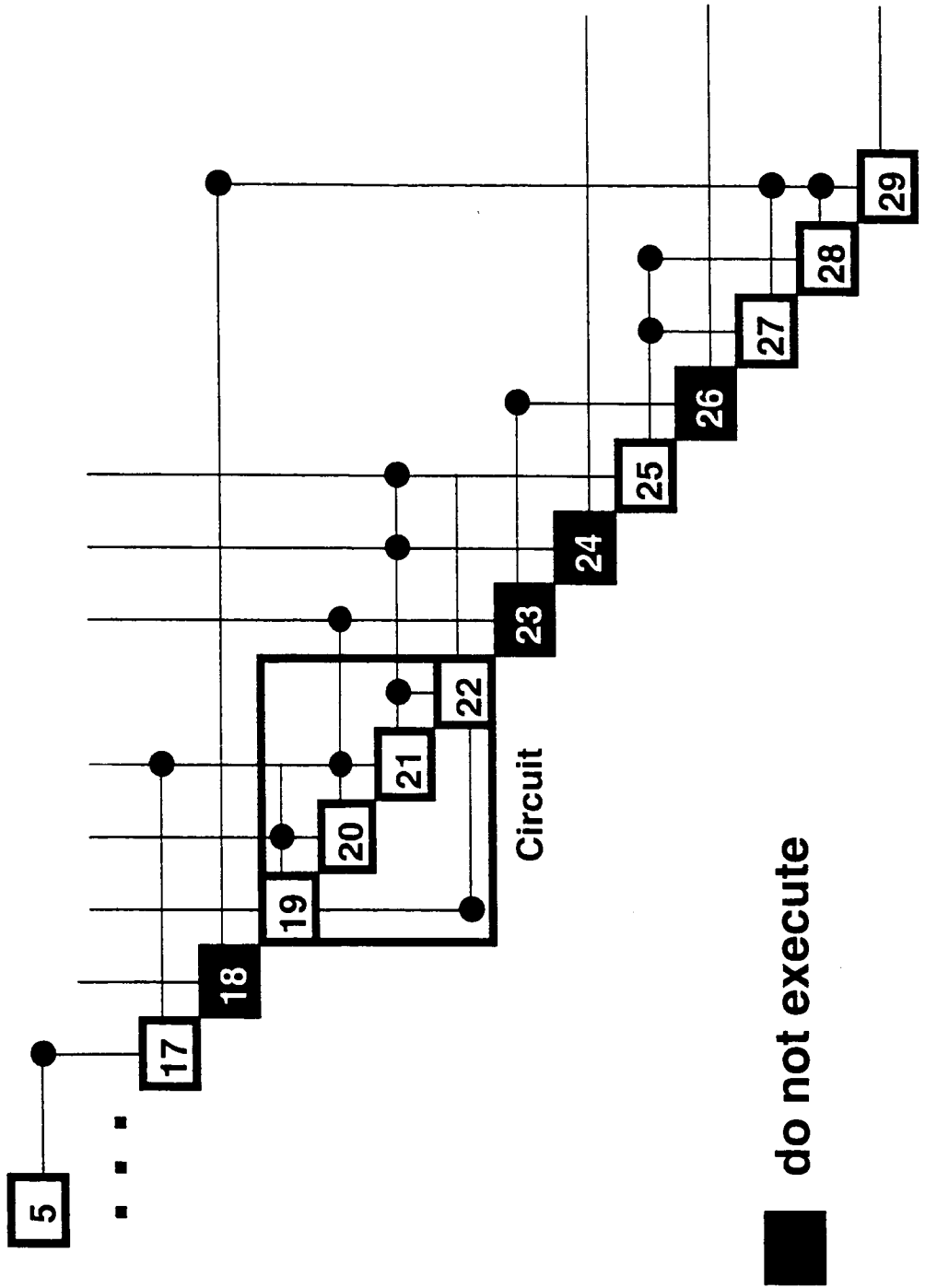


A CHANGE IS MADE

**A change is made in the variable
gross weight - module 5 (input data)**

**What modules must be re-executed to determine
the effect this will have on the variable
elastic polar curves - module 29**

MODIFIED DESIGN STRUCTURE MATRIX



SELECTING MODULES FOR RE-EXECUTION

The following modules must be rerun:

TPWEIGHTb (17) - fuel weight and concentrated weight

--- loop --- SUPERPOS (19) - pressure distribution

--- loop --- TRIM (20) - load distribution

--- loop --- ELAPSB (21) - wing deformations

--- loop --- CAMBER (22) - pressure due to deformations

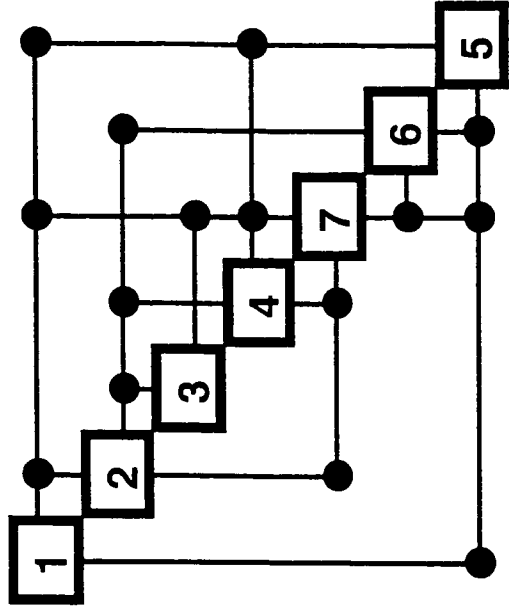
WDMOD3a (25) - aircraft geometry with elastic deformations

WINGDES (27) - induced drag

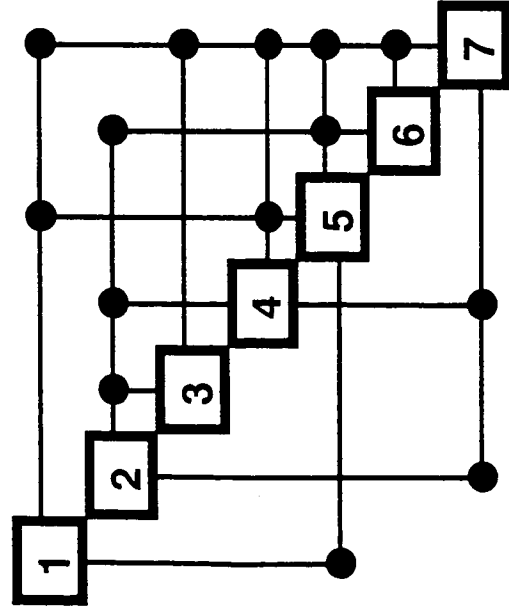
AWAVE (28) - wave drag

POLAR (29) - elastic polar curves

EXAMINING DESIGN TRADE-OFFS WITH DEMAND



**6 feedbacks
no crossovers**



**3 feedbacks
with crossovers**

NEW CAPABILITIES

Determine "weak" and "strong" couplings

Optimize the ordering of modules within a circuit

Breakdown of output values

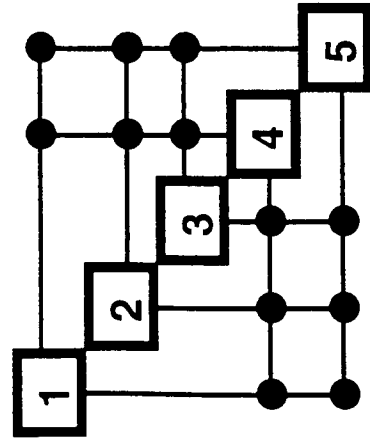
STRENGTH OF COUPLINGS

Determine strength of couplings through sensitivity analysis (Bloebaum SUNY Buffalo)

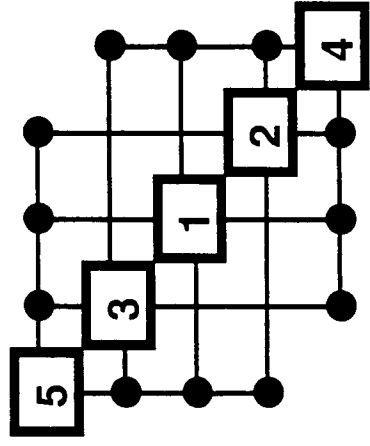
Suspend these couplings (remove process or coupling) without sacrificing system accuracy

Develop rules to determine relationship between a module and those modules coupled to it

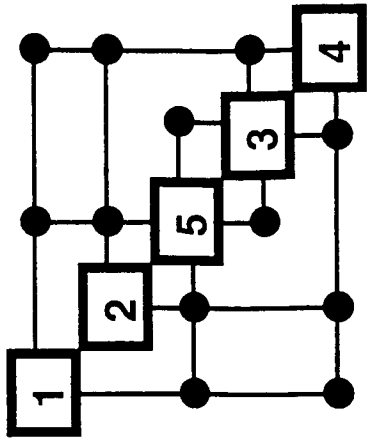
OPTIMIZE THE ORDERING WITHIN A CIRCUIT



Option 1



Option 2



Option 3

Timings

Module 1 - 5	Module 2 - 10	Module 3 - 15
Module 4 - 20	Module 5 - 25	

OPTIMIZE THE ORDERING WITHIN A CIRCUIT

$$\text{Cost function} = c1 * \# \text{ feedbacks} + c2 * \# \text{ crossovers} + c3 * \text{timing function} + c4 * \text{sensitivity function}$$

**where c's are user-defined
weight coefficients**

TIMING FUNCTION

Count number of times a module appears in an iteration

Multiply number of appearances by the execution time of the module

Sum to find the total time

Option 1 = 335 Option 2 = 255 Option 3 = 295

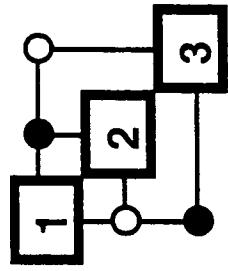
Option 2 has crossovers

SENSITIVITY FUNCTION

Strong coupling = 1 and weak coupling = 0
 (actually there will be a wider range)

Add the numbers for feedforward

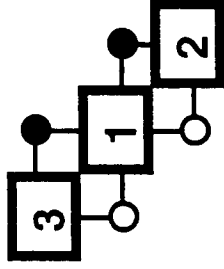
Subtract the numbers for feedback



Option 1

- Strong couplings (1)
1 to 2 and 3 to 1

- Weak couplings (0)
1 to 3 and 2 to 1



Option 2

$$\text{Option 1} = (1+0) - (1+0) = 0$$

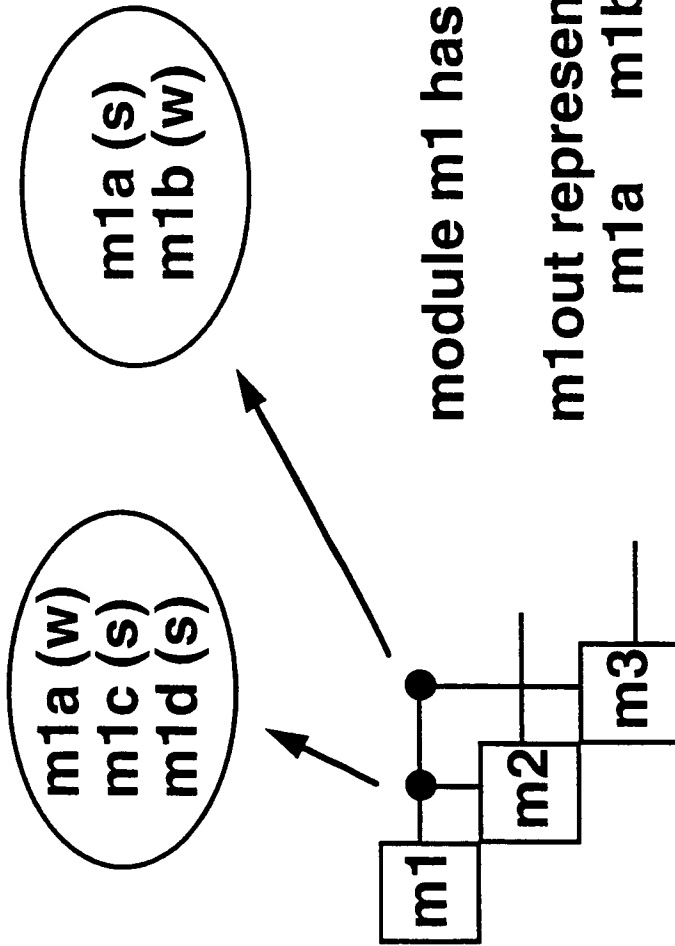
$$\text{Option 2} = (1+1) - (0+0) = 2$$

BREAKDOWN OF OUTPUT

Current approach uses only a single name to represent the output of a module

New approach allows single name to reference several output names

EXAMPLE



module m1 has output name m1out

m1out represents four outputs:

m1a m1b m1c m1d

SUMMARY

DeMAID is a knowledge-based tool developed to aid the design manager in understanding a complex design problem

DeMAID is available at Georgia Tech, MIT, SUNY Buffalo, Boeing, General Electric, Northrop, McDonnell Douglas, DEC, Lockheed Ft. Worth, CERC, and Rockwell (widespread interest)

New capabilities to include more optimum sequencing and more coupling information

Several areas for research in design

RDD-100 and the Systems Engineering Process

Robert D. Averill, Systems Engineering Office, AMSD, IOG

P.26

Efforts to implement an effective systems approach to NASA programs are in progress Agency-wide ¹. At Langley we are trying to define an enhanced systems engineering process for in-house flight projects to assure that each system will achieve its goals with quality performance and within planned budgets and schedules. An effective systems engineering approach applied throughout the project life cycle can help Langley produce a better product. This paper will show how this can be done by utilizing a systems engineering process in combination with available software tools such as RDD-100 ². To accomplish this, I will, first, briefly discuss the systems engineering process and then show how RDD-100 has been applied as a pilot effort in the early phases of the SABER ³ instrument development.

(Chart 2) The objective is to show you how RDD-100 can be used as a systems engineering tool throughout the project life cycle and to challenge you to consider using this tool with your project team.

(Chart 3) Systems engineering may mean many different things to different people but this is the way it is defined in the Langley Systems Engineering Handbook ⁴ which is currently pending publication. The systems engineering process is really the key to how we approach the problem. There are many different procedures, methodologies, and models being used for systems engineering. It is important that each project define how systems engineering will be managed and conducted throughout the project life cycle.

(Chart 4) The Systems Analysis and Design Procedure is proposed for use at Langley during the Formulation Phases of the project when the systems engineering activity is the most intensive. This procedure provides a focused and structured systems engineering method and is a problem solving approach which can be tailored to project needs.

(Chart 5) The Systems Analysis and Design Procedure is a ten step process applied iteratively during each phase of the project. The concentric circles represent each phase; for example, the inner circle symbolizes the Pre-Phase A effort which has the purpose of quickly assessing the feasibility of a proposed project to determine if it justifies further development. The detailed activities of each step of the process are developed in more detail in LHB 7122.1. However, they can be quickly summarized as follows. The Initialization step includes a management decision to initiate the study and provide skills and resources necessary to do the job on a timely basis. The determination of User Needs and Goals is perhaps the most important step in scoping the effort; this leads directly to a definition of Systems Requirements to achieve the goals. Performance Measures are defined to provide a quantitative standard to assess system performance. Next, potential System Concepts are generated, Analyzed, and Ranked to determine system feasibility. Further Systems Development may be needed to bring the proposed system approaches to the level of maturity desired for this initial stage of development. The final step in the process provides for technical and management reviews to assess the status of the development. This represents a Decision Point which will determine if

the system will repeat the iterative development process or pass to the next phase of project development. It is believed that the use of such a customized systems engineering process with well defined tasks, products, and controls will help the Project Team perform most effectively. It should be emphasized that the systems engineering process is a team effort and is dependent upon project teamwork and communication throughout the process.

(Chart 6) The goal of the process is to enhance communication between different technical disciplines on the project. For example, the relationship between systems engineering and software engineering is vital to the success of the project. These two groups must work closely together to define their mutual information needs. Are the typical systems engineering "products" in the left column useful to the software engineering function? Are the typical software engineering "products" in the right column a logical and related extension of the systems engineering requirements? The project can operate most efficiently if a common technical language is used by all of the project team.

(Chart 7) There are currently available several computer aided systems engineering tools which propose to provide a common technical language for use throughout the project life cycle. One of these is the Object Modeling Technique developed by General Electric ⁵ and currently being marketed as StP/OMT ⁶. This tool is being evaluated for use at Langley but is not currently implemented. The RDD-100 tool is being used in the Systems Engineering Office at Langley. RDD-100 utilizes an object oriented methodology with a symbolic language designed to be useful to all technical disciplines.

(Chart 8) RDD-100 is an extension of the earlier Entity-Relationship Model (developed originally for information modeling use ⁷) into an object modeling concept. The power of the RDD-100 concept is that the Elements (Entities) are linked by binary relationships such that changes to any element are transferred to its related Elements; thus continuously updating the database. The tool also provides for requirements tracking throughout the system life cycle. Another powerful feature of RDD-100 is its modeling capability.

(Chart 9) The Integrated System Model is an evolutionary development which begins with the most rudimentary concept of system objects and progressively evolves into a complete model representing overall system dynamic performance. This provides continuity through the project life cycle and offers a "seamless" transition from phase-to-phase.

(Chart 10) We will now present a brief overview of RDD-100 capabilities.

(Chart 11) RDD-100 is a menu driven application and provides ready access to all of its features. It can be seen that the emphasis of the program is on system Elements. The Multi-Element View and the various Editors permit easy manipulation and editing of the system Elements.

(Chart 12) An example of the Multi-Element View concept is the SABER Requirements hierarchy. The Element-Relationship aspect is shown as, for example, Operational Objective: Interface Constraints incorporates Operational Objective: Instrument Mass. ~

(Chart 13) Shown here is a section of the requirements Custom Hierarchy which provides a visual display of the relationships between requirements.

(Chart 14) The modeling capability of RDD-100 is implemented by Behavioral Diagrams which incorporate all of the system functional and dynamic relationships on one diagram. This is a major advantage over other concepts which separate, for example, control and data functions on two unsynchronized models. The RDD-100 approach provides one self-contained Integrated System Model which demonstrates system dynamic response.

(Chart 15) This is the overall SABER Operational Model based on the five key objects selected for the system: User, Ground Station, Spacecraft, SABER Instrument, and Atmospheric Scene. The purpose of this Operational Model is to demonstrate the flow of top level control and data messages.

(Chart 16) The SABER instrument is shown here in more detail with the operational functions of current interest. The behavioral diagram includes Time Functions, Time Items, and, in this case, an Iterate Function, represented by the loop, which repeats the scan sequence for a specified number of cycles.

(Chart 17) The scenario shown is a running model and can be evaluated by the Dynamic Verification Facility. The system runs on an arbitrary time base which can represent any desired time scale. Various functions can be selected to display an Events Transcript, Time Lines, and System Resources. The Facility identifies any dynamic inconsistencies in the model.

(Charts 18 & 19) Shown are sections of the Event Transcript showing the beginning and end of the run.

(Charts 20 and 21) Shown are the Function Time Line and a history of the Scene Radiance resource. The instrument, in this example, accumulates ten data samples and then transmits them to the spacecraft.

(Charts 22 & 23) The Summary concludes that the use of a structured systems engineering process in conjunction with a powerful computer aided systems engineering tool is believed to provide the most effective approach to achieving project success at LaRC.

- 1 NASA Systems Engineering Handbook, Draft, September 1992, JPL.
- 2 RDD-100 - Requirements Driven Development, Ascent Logic Corporation.
- 3 Sounding of the Atmosphere using Broad band Emission Radiometry.
- 4 LHB 7122.1, Systems Engineering Handbook for In-House Space Flight Projects.
- 5 Rumbaugh, James; et al: Object Oriented Modeling and Design. Prentice Hall, 1991.
- 6 STP/OMT - Software through Pictures/ Object Modeling Technique, Martin Marietta Advanced Concepts Center and Interactive Development Environments.
- 7 Sage, Andrew P., and Palmer, James D.: Software Systems Engineering. John Wiley and Sons, 1990.

**RDD-100 and the
Systems Engineering Process**

The Role of Computers in LaRC R&D

June 16, 1994

Robert D. Averill

RDD - 100
Requirements Driven Development

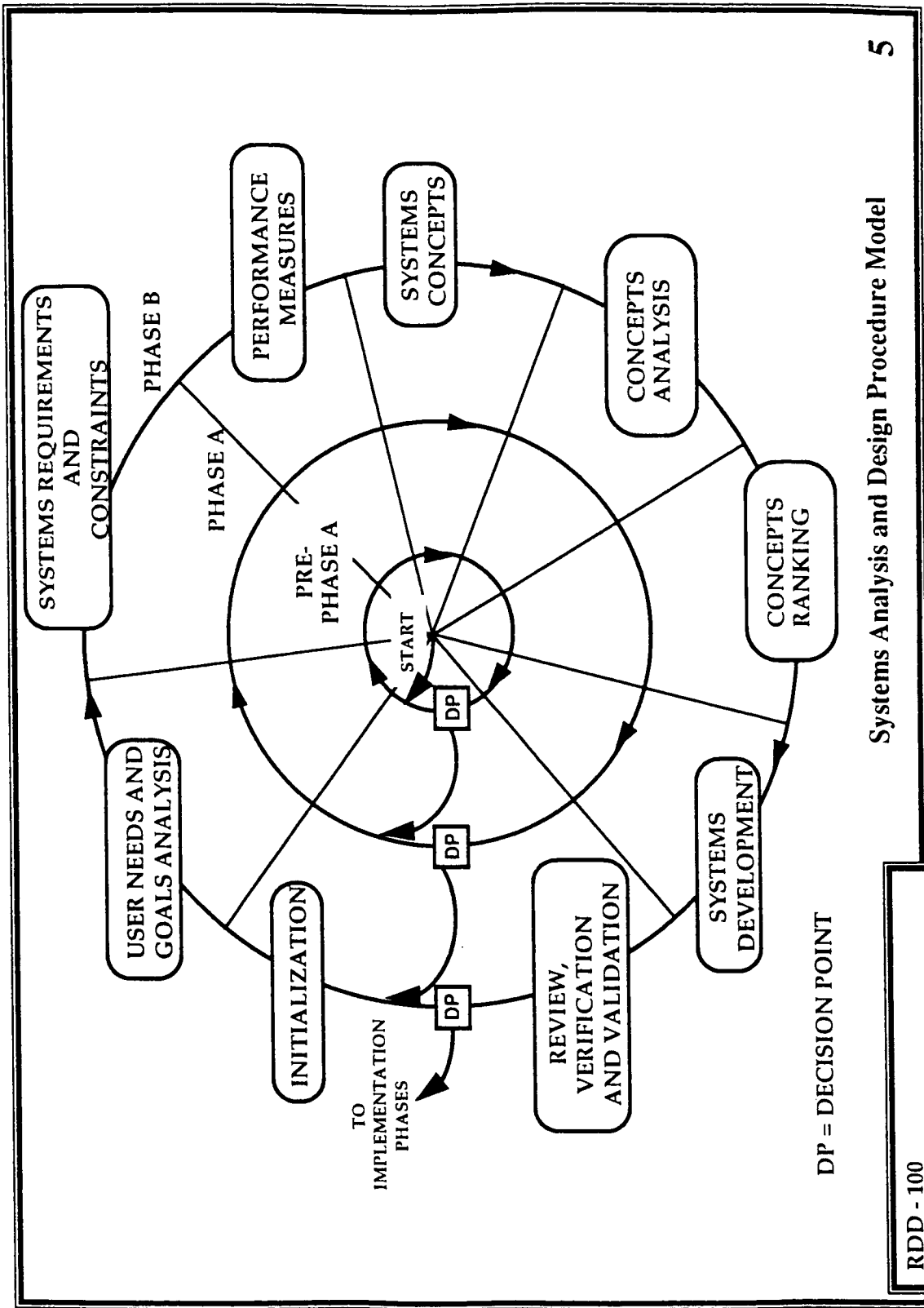
OBJECTIVE

- How RDD-100 can be used as a Systems Engineering Tool throughout the project life cycle.

- **Systems Engineering:** A function that guides the transformation of customer/user's needs into a flight system that meets technical performance requirements. . .
- The **objective of systems engineering** is to provide a robust system which satisfies the customer's technical performance objectives within the constraints of cost and schedule.
- The **systems engineering process** is the approach to achieve this objective and is defined in the Langley Systems Engineering Handbook (LHB 7122.1)

- **Systems Analysis and Design Procedure:** An interactive, analytical, step-by-step approach during the Formulation Phases of the project life cycle.

- Pre-Phase A: Preliminary requirements and concepts analysis.
- Phase A: Requirements definition and conceptual trades.
- Phase B: Concept definition and preliminary design.



Systems Analysis and Design Procedure Model

Bridging the Gap

(Need for Common Terminology)

Systems Engineering

- Goals Analysis
- System Requirements
- System Concepts
- Project Database
- Design Specifications
- System Model
- Performance Verification Matrix
- Other _____



Software Engineering

- Operations Model
- Requirements Model
- Control/Data Flow Diagrams
- System Architecture
- Methodologies
(SA/SD, JSD, OOA, ESP)
- V & V; Risk Analysis
- Data Dictionary; P-Specs
- Other _____

RDD - 100
Requirements Driven Development

RDD - 100 Initiatives

- RDD - 100 is a product family developed by Ascent Logic to support the systems engineering process and improve program success.
- RDD - 100 objective is to enable concurrent engineering methodologies and support communication between engineering disciplines.
- RDD - 100 utilizes an object-oriented methodology with “bridges” to CASE tools such as Teamwork and Software through Pictures (StP)

RDD - 100 Systems Engineering Improvements

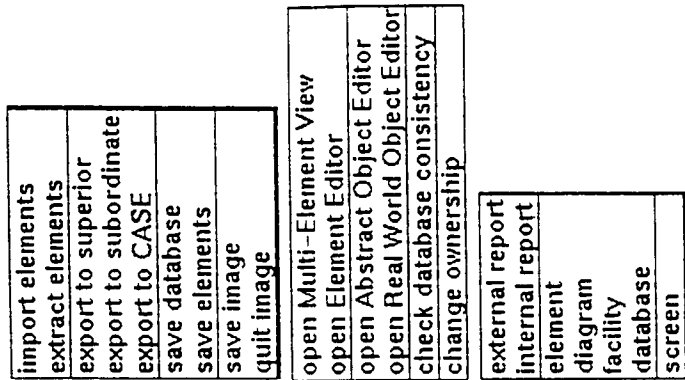
- Developing an executable notation for systems definition
 - Elements - class of objects
 - Relationships - links to other elements
 - Attributes - unique characteristics
- Maintain an interrelated database to record traceability and design decisions.
- Use of a systematic and automated documentation process.
- Requirements extraction, definition, allocation, and tracking.
- Increased use of modeling and prototyping to identify timing and interface issues.

Integrated System Model Provides An Evolutionary Modeling Representation Of The System

- Real World Object Model - system context and "objects".
- Data Model - external message flows.
- Operational scenarios - stimulus and response sequence of system elements.
- Conceptual Behavior Model - system model defining system objects and functions.
- Component Interconnection Model - defines system requirements allocations.
- Stimulus/Response Model - overall dynamic performance.

RDD - 100 Walk - Through

- Main Menu
- SABER Requirements.
- Hierarchy of Requirements.
- System Engineering Notebook → Report Writer.
- SABER System Scenario.
- Operational Model → Dynamic Verification Facility.



The RDD - 100 System Designer Main Menu

RDD - 100
Requirements Driven Development

Source: NASA2.txt

NUMBER.

DESCRIPTION. The overall goal of the SABER experiment is to improve understanding of the thermal, chemical, and dynamical structure and the energetics of the mesosphere and lower thermosphere by conducting global-scale measurements of kinetic temperature, radiatively and chemically significant minor species concentrations, and long-lived species for use as dynamical tracers.

[1] documents OperationalObjective: Infrared Remote Sensing

NUMBER. 1.0

DESCRIPTION. To apply the space-proven technique of infrared earth limb emission remote sensing to the virtually unexplored region above ~ 50 km.
Ref. Proposal par. 2.1, 4.0.

[2] incorporates OperationalObjective: Interface Constraints

NUMBER. 1.1

DESCRIPTION. To conform to the interface constraints of the TIMED spacecraft.

[3] incorporates OperationalObjective: Instrument Mass

NUMBER. 1.1.1

DESCRIPTION. To develop an instrument of <66 kg mass.

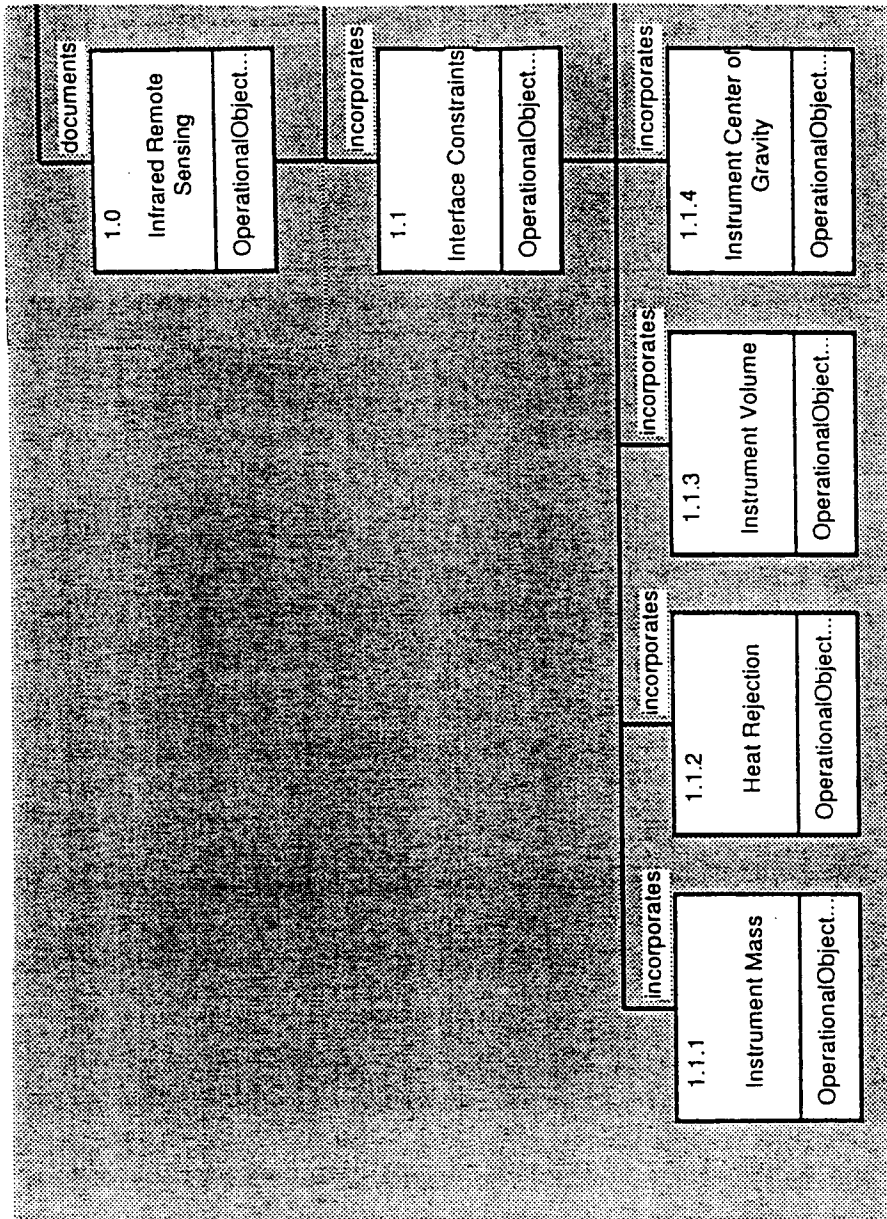
[3] incorporates OperationalObjective: Heat Rejection

NUMBER. 1.1.2

DESCRIPTION. To develop an instrument requiring <50 watts of heat rejection to the spacecraft at a temperature of <290 K.

SABER Requirements

RDD - 100
Requirements Driven Development

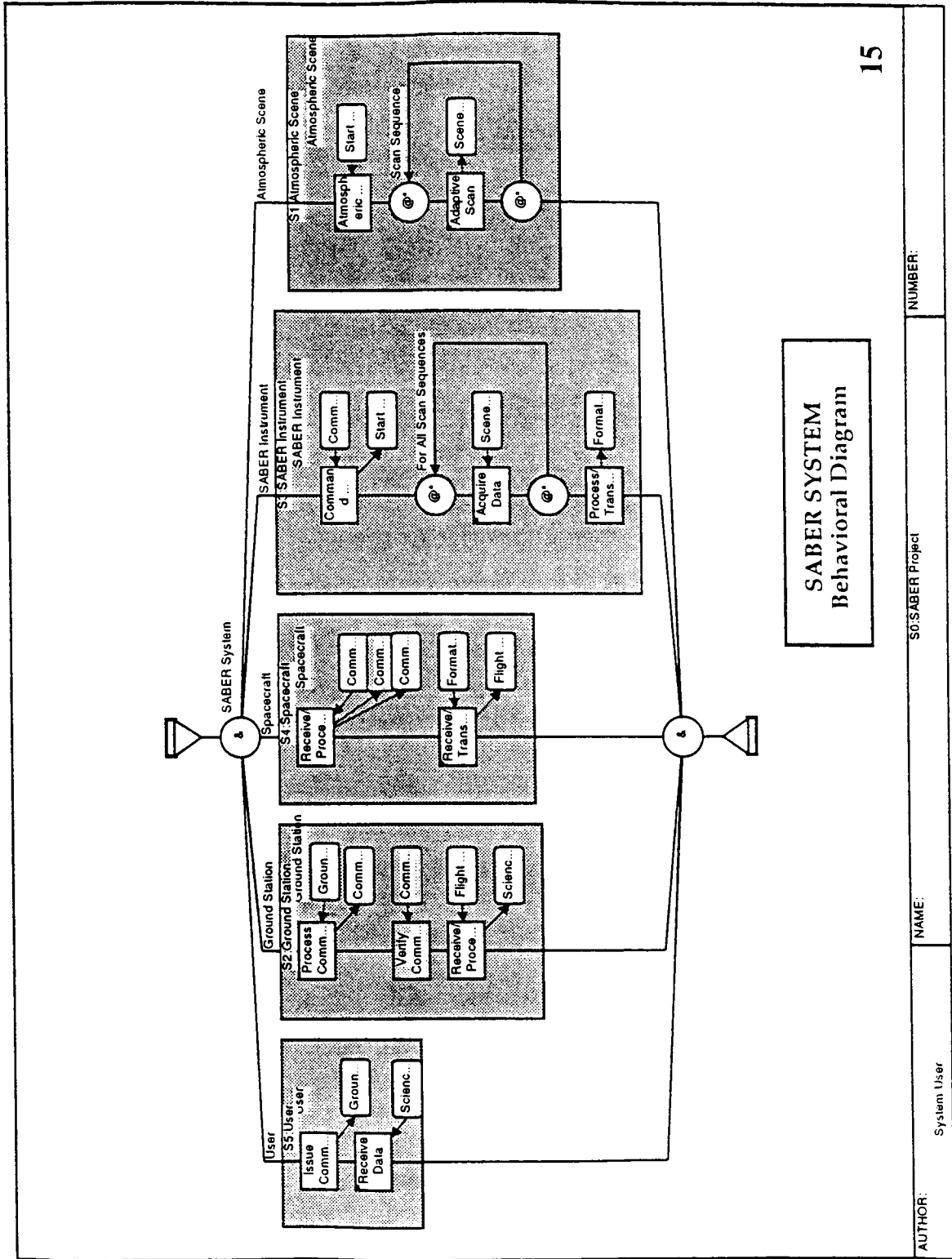


SABER Requirements Hierarchy

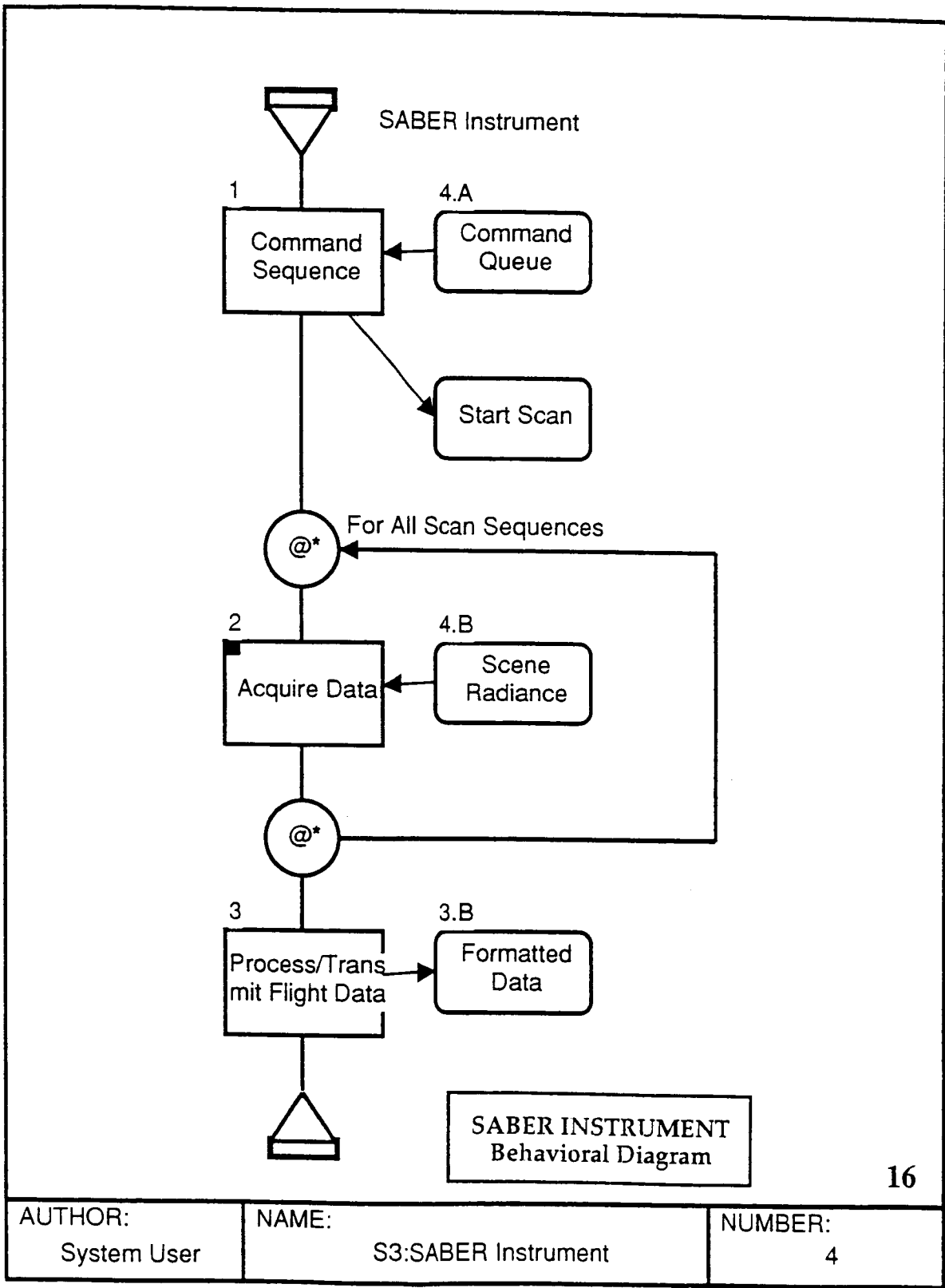
RDD - 100
Requirements Driven Development

RDD - 100 Implemented by Behavioral Diagrams

- Defines system scenarios and interfaces.
- Verifies functionality and performance.
- Provides an Integrated System Model.
- Demonstrates system dynamic response.

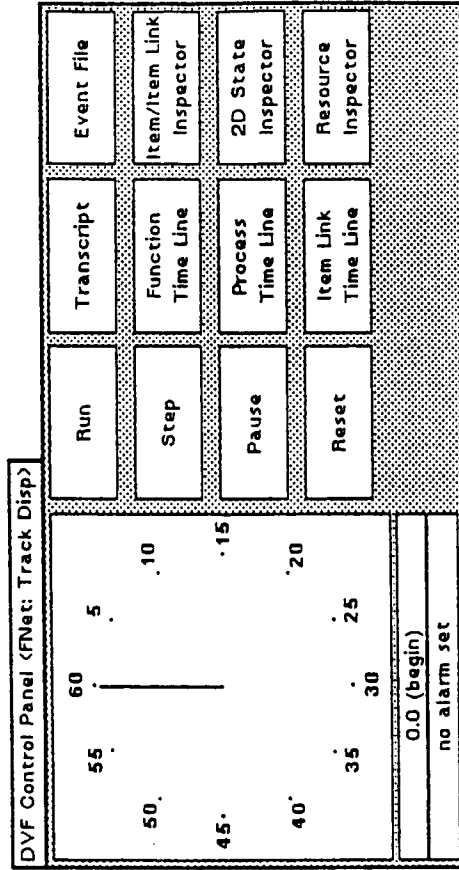


SABER SYSTEM Behavioral Diagram



AUTHOR: System User	NAME: S3:SABER Instrument	NUMBER: 4
------------------------	------------------------------	--------------

DVF Control Panel



SABER Dynamic Verification Facility Control Panel

RDD - 100
Requirements Driven Development

Event Transcript

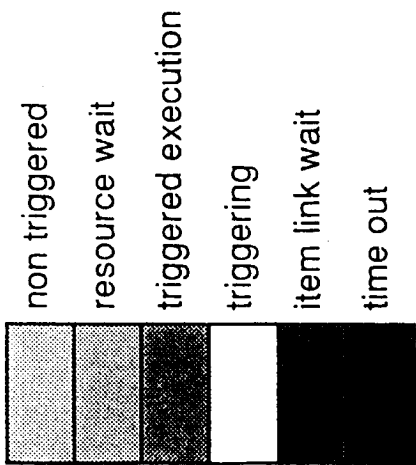
0.0 [S0:SABER Project] created
0.0 [S0:SABER Project] creating process ('User')
0.0 [S0:SABER Project] creating process ('Ground Station')
0.0 [S0:SABER Project] creating process ('Spacecraft')
0.0 [S0:SABER Project] creating process ('SABER Instrument')
0.0 [S0:SABER Project] creating process ('Atmospheric Scene')
0.0 [User] created
0.0 [Ground Station] created
0.0 [Spacecraft] created
0.0 [SABER Instrument] created
0.0 [Atmospheric Scene] created
0.0 [User] (TimeFunction: Issue Commands) enabled
0.0 [User] (TimeFunction: Issue Commands) proceeding
0.0 [User] (TimeFunction: Issue Commands) working (10.0 10.0)
0.0 [Ground Station] (TimeFunction: Process Command) enabled
0.0 [Ground Station] (TimeFunction: Process Command) waiting for message
0.0 [Spacecraft] (TimeFunction: Receive/Process Commands) enabled
0.0 [Spacecraft] (TimeFunction: Receive/Process Commands) waiting for message
0.0 [SABER Instrument] (TimeFunction: Command Sequence) enabled
0.0 [SABER Instrument] (TimeFunction: Command Sequence) waiting for message
0.0 [Atmospheric Scene] (TimeFunction: Atmospheric Radiance) enabled
0.0 [Atmospheric Scene] (TimeFunction: Atmospheric Radiance) waiting for message
10.0 [User] (TimeFunction: Issue Commands) sending message out to ('Timeltem' 'Ground Commands'
nil nil 0 0 'Ground Station' 4067)
10.0 [User] (TimeFunction: Issue Commands) ended
10.0 [User] (TimeFunction: Receive Data) enabled
10.0 [User] (TimeFunction: Receive Data) waiting for message
10.0 [Ground Station] (TimeFunction: Process Command) received message ('Ground Commands' nil
'User' 4067)

SABER Event Transcript (Start)

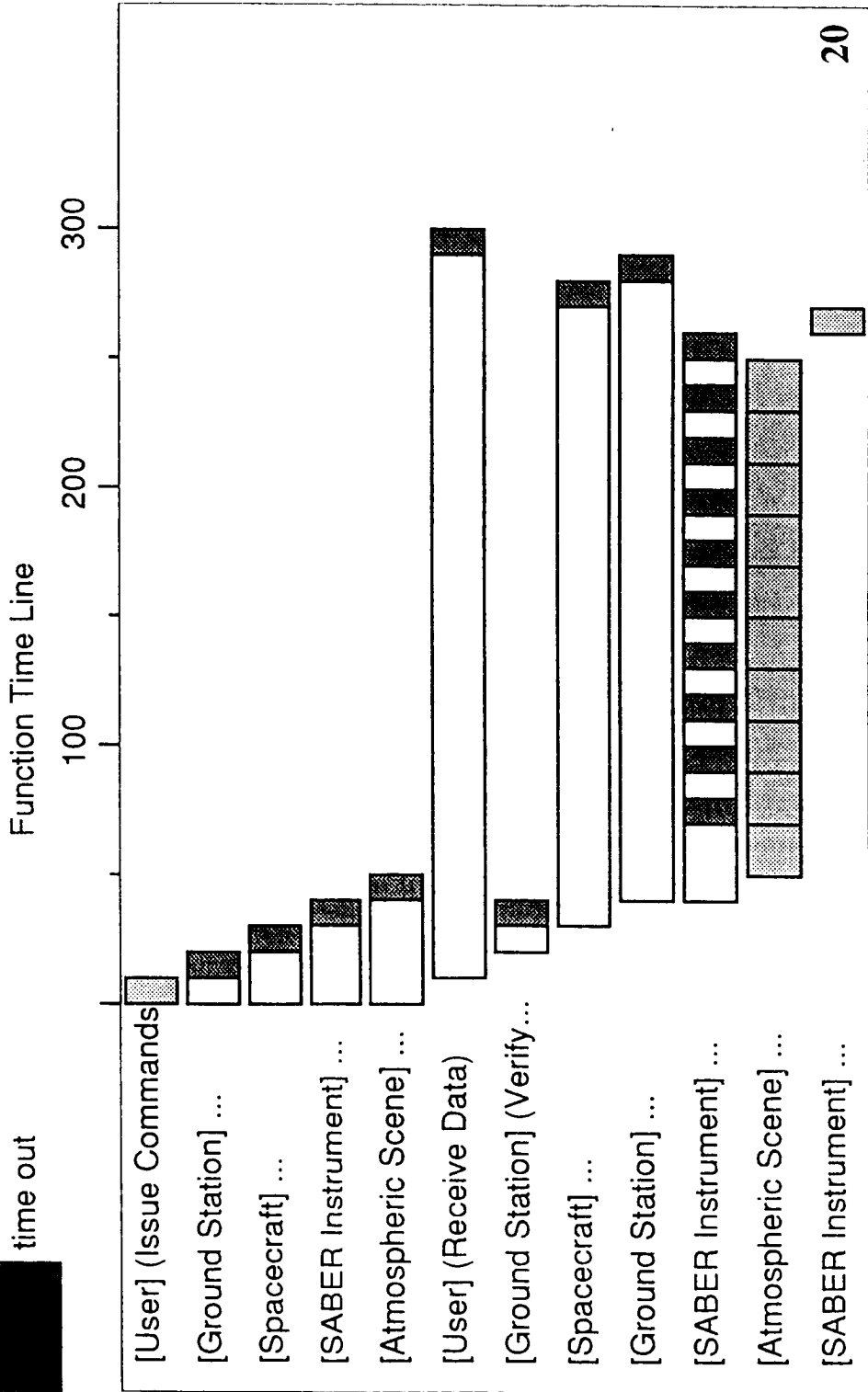
Event Transcript

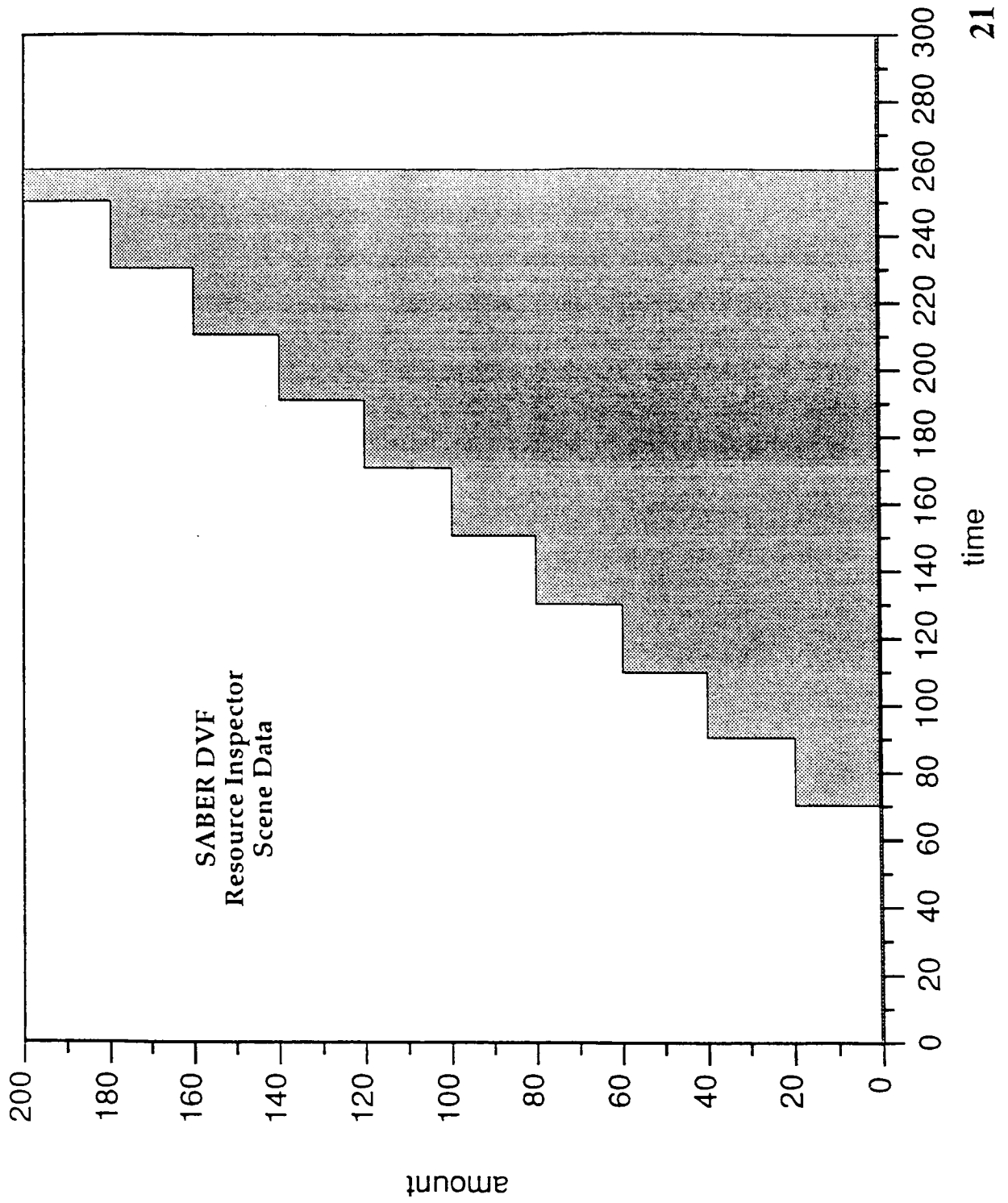
280.0 [Ground Station] (TimeFunction: Receive/Process Flight Data) consuming resource ('Resource'
'Flight Data' 200.0 200.0)
280.0 [Ground Station] (TimeFunction: Receive/Process Flight Data) resources received ('Resource'
'Flight Data' 200.0 0.0)
280.0 [Ground Station] (TimeFunction: Receive/Process Flight Data) working (10.0 290.0)
290.0 [Ground Station] (TimeFunction: Receive/Process Flight Data) producing resource ('Resource'
'Science Data' 200.0 0.0)
290.0 [Ground Station] (TimeFunction: Receive/Process Flight Data) sending message out to ('TimeItem'
'Science Data' nil nil 0 'User' 980)
290.0 [Ground Station] (TimeFunction: Receive/Process Flight Data) ended
290.0 [Ground Station] terminated
290.0 [User] (TimeFunction: Receive Data) received message ('Science Data' nil 'Ground Station' 980)
290.0 [User] (TimeFunction: Receive Data) triggered
290.0 [User] (TimeFunction: Receive Data) proceeding
290.0 [User] (TimeFunction: Receive Data) consuming resource ('Resource' 'Science Data' 1.0 200.0)
290.0 [User] (TimeFunction: Receive Data) resources received ('Resource' 'Science Data' 1.0 199.0)
290.0 [User] (TimeFunction: Receive Data) working (10.0 300.0)
300.0 [User] (TimeFunction: Receive Data) ended
300.0 [User] terminated
300.0 [SO:SABER Project] terminated

SABER Event Transcript (End)



SABER DVF
Function Time Line





Summary

- The Langley Systems Engineering Process provides a well defined and effective approach to systems engineering.
- A Langley Systems Engineering Handbook (LHB 7122.1) will allow Project Managers and Systems Engineers to tailor the systems engineering process for their projects.
- The use of computer aided systems engineering tools will allow project teams to operate more productively.

Summary

RDD - 100 provides:

- A powerful tool for improving the systems engineering process.
- An interrelated database for the management of project requirements.
- A method for rapid simulation and modeling of system performance.
- A common technical language to improve communication between technical disciplines.
- A method for automated documentation and direct requirement linkage on projects.

356138

110057

N95-16472

COMPUTER TOOLS FOR SYSTEMS ENGINEERING AT LARC

P. 64

J. Milam Walters, Systems Engineering Office
Aerospace Mechanical Systems Division

The Systems Engineering Office (SEO) has been established to provide lifecycle systems engineering support to LaRC projects. Over the last two years, the computing market has been reviewed for tools which could enhance the effectiveness and efficiency of activities directed towards this mission. A group of interrelated applications have been procured, or are under development including a requirements management tool, a system design and simulation tool, and a project and engineering database. This paper will review the current configuration of these tools and provide information on future milestones and directions.

The Role of Computers In LaRC R&D

Computer Tools for Systems Engineering

Presented by

J. Milam Walters

June 16, 1994

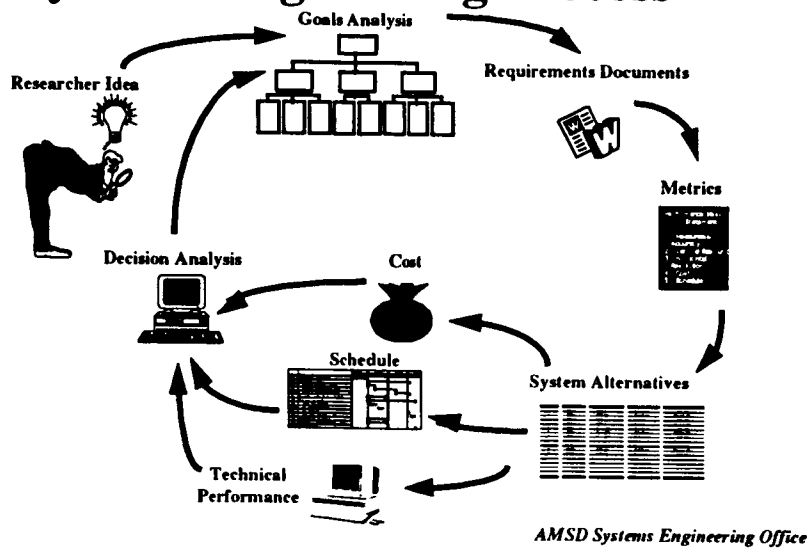
AMSD Systems Engineering Office

AMSD Systems Engineering Office

- **Established via Center Reorganization after approximately 3 years of ground laying**
- **Current staffing level - 5 CS**
- **Chartered to guide application of systems engineering to LaRC flight projects**
- **Process detailed in LHB 7122.1, currently in approval cycle**
- **Process applied to various projects, most recently JADE and SABER**

AMSD Systems Engineering Office

Systems Engineering Process



Systems Engineering Office Tools

- Workstation Based tools consist of the following:
 - Oracle SE Project & Engineering Database
 - Excalibur Scanning/Recognition Software
 - RTM (Requirements Traceability & Management)
 - RDD-100 (Requirements Driven Design)
 - Interleaf 6.0
 - Matlab with following toolboxes:
 - Simulink option
 - System Identification
 - Control System
 - Optimization

AMSD Systems Engineering Office

Oracle SE Project & Engineering Database

- **Oracle RDBMS Version 7** relational database on SUN Sparcstation 10, Model 41
- **Oracle*Forms Version 4** provides graphical user interface for record & graphic viewing
- **Oracle Data*Query** performs complex searches
- **Database consists of pre-set form types, with the capability to quickly generate any new table type**
- **Database will store project documentation and graphics as well as engineering data tables**

AMSD Systems Engineering Office

Excalibur Scanning/Recognition Software

- **Interfaces with document scanner to read and interpret input**
- **Contains an adaptive search engine to retrieve desired document**
- **Displays original image upon match of a given search**
- **Provides the capability to scan and store input documents**

AMSD Systems Engineering Office

RTM Requirements Traceability & Management

- **Application developed especially for tracking and managing project requirements**
- **Utilizes Oracle database to store requirement information**
- **Provides special tools for:**
 - extracting requirements from source documents
 - expanding and focusing requirements
 - general requirements maintenance
- **Includes output bridges to RDD-100 and Interleaf**

AMSD Systems Engineering Office

RDD-100 Requirements Driven Design

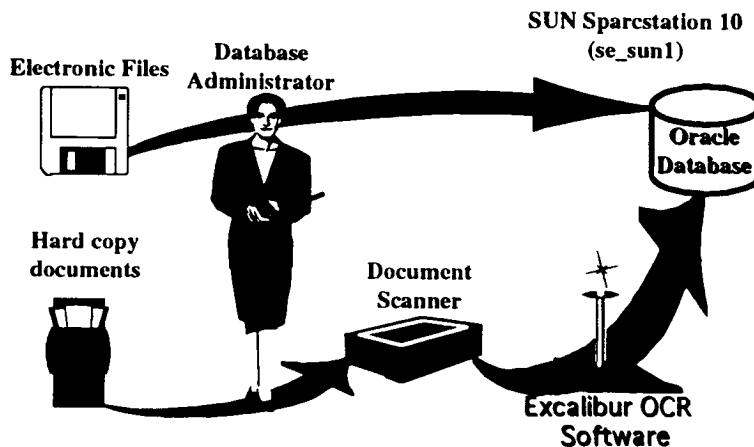
- **Facilitates the construction, maintenance, display, and documentation of design objects that specify behavior**
- **Objects are created and edited by graphics or text, with multiple generated views available to gain different perspectives**
- **Includes a simulator which directly executes the design objects**
- **Templates and consistency checks verify system design sufficiency**
- **Bridge to Interleaf is included**

AMSD Systems Engineering Office

Matlab

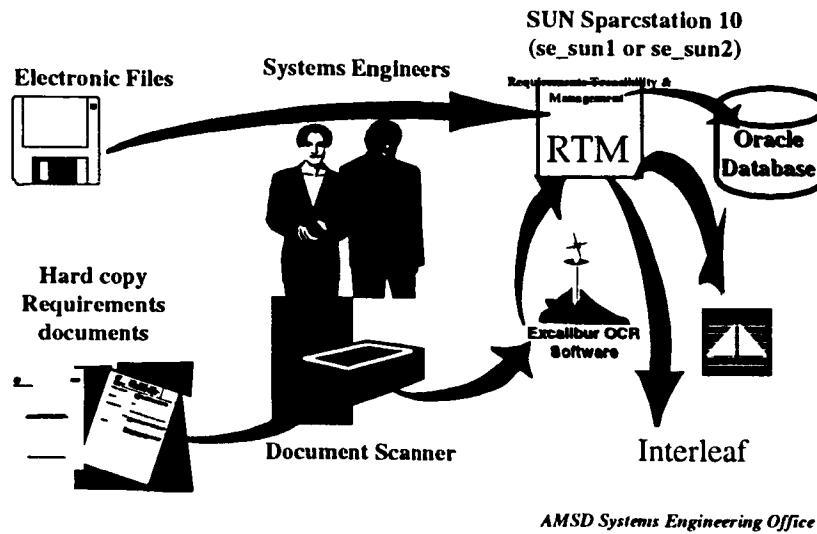
- Interactive software program for scientific and engineering numeric computation
- Combines numerical analysis, matrix computation, signal processing, and graphics with a user interface through standard math notation.
- Functions include differential equation solution, polynomial operations, matrix computation, complex arithmetic and signal processing tools
- To view data graphically, MATLAB provides 2-D linear, log, semilog, and polar plots, and 3-D mesh and contour graphs
- Works with MATLAB numeric computation software package to build mathematical models of

Database Population

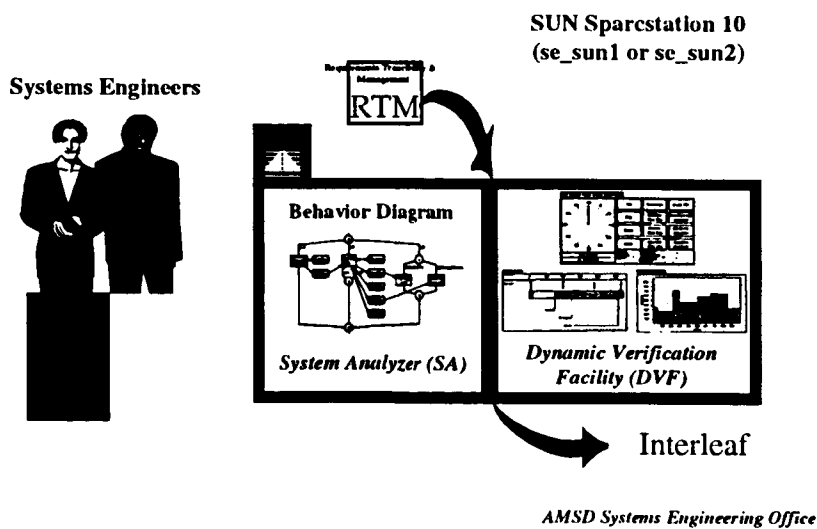


AMSD Systems Engineering Office

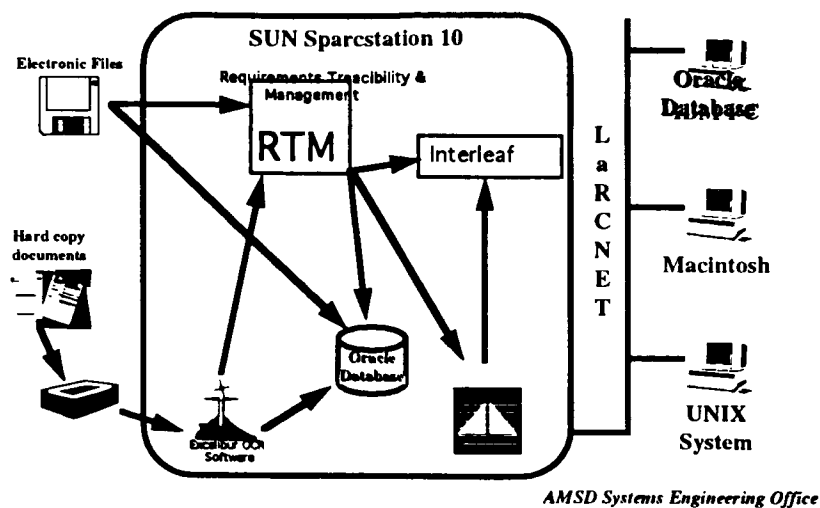
Requirements Management



System Modeling & Simulation



Tool Interface Overview



Summary

- The Systems Engineering Office of AMSD has been established to provide for computer aided:
 - systems level behavior modeling and simulation of new concepts (RDD-100)
 - subsystem mathematical modeling and simulation (Matlab)
 - requirements tracking and management (RTM)
 - storage of project and engineering documentation (SEDB)
- Interested parties should contact Richard Foss at 4-7049 or Milam Walters at 4-3014

AMSD Systems Engineering Office

A Distributed Computing Environment for Multidisciplinary Design

The Framework for Interdisciplinary Design Optimization (FIDO) project has the goal of developing a general distributed computing system for executing multidisciplinary computations on a networked heterogeneous cluster of workstations and vector and massively parallel computers. This project is a part of the Computational Aerosciences (CAS) project in the High Performance Computing and Communications (HPCC) program. The FIDO system provides a means for automating the total design process. It facilitates communication and control between components of the system, which include the diverse discipline computations involved in a design problem and the system services that facilitate the design. In its current state of development, the prototype FIDO system is being applied to a token example of the optimized design of a high-speed civil transport (HSCT), involving a simplified problem that includes the disciplines of aerodynamics, performance, propulsion, and structures, but with very few design variables. However, it has already demonstrated the ability to coordinate multidisciplinary computations and communications in a heterogeneous distributed computing system.

The concept being used in FIDO is course-grained parallelism, with instances of the disciplinary codes (aero, structures, etc.) running on separate processors, under control of an executive on another processor, and exchanging data through a single data base manager (on yet another processor). To allow the user to monitor the progress of the design iterations, there is a graphical user interface (which tracks the execution of codes performing the design iterations) and a separate process, called SPY, which allows its user to extract and plot data produced during current and previous design cycles. In fact, multiple instances of SPY can be executing at once, so that the designer can call on discipline experts and they (possibly from some remote location on the Internet) can examine the results being produced and provide advice. SPY is currently being upgraded to provide the capability for the designer to make changes in variable values during execution and so guide the design process.

The distributed computing system currently includes Sun, Silicon Graphics, and Digital Equipment Co. workstations. Provision has been made for adding connections to Cray vector computers and Intel parallel computers, and preliminary checks of connection procedures have been carried out. A communications library has been written (implemented using the PVM basic library developed at Oak Ridge National Lab) to provide the versatility for transferring data packages ranging from single variables or file names to large data arrays.

The current Motif-based Graphical User Interface (GUI) consists of three separate elements: setup, application status, and data display. The setup GUI provides the user with a convenient means of choosing the initial design geometry, material properties, and run conditions from a pre-defined set of files. The interface displays the choices using a series of pop-up Motif data windows, and allows the user to modify and store new condition files. The application status GUI allows the user to monitor the status of a design run. An example of this display is shown in the slide entitled "FIDO User Interface Concept". Within the screen on the left part of the slide, the upper left window displays current run parameters and contains pull-down menus for setting various options. The right window graphically displays the state of the overall design process by changing the color of each labelled box according to the work being done. A color key is shown in the lower left window. Additional detail of the system state can be obtained by selecting the

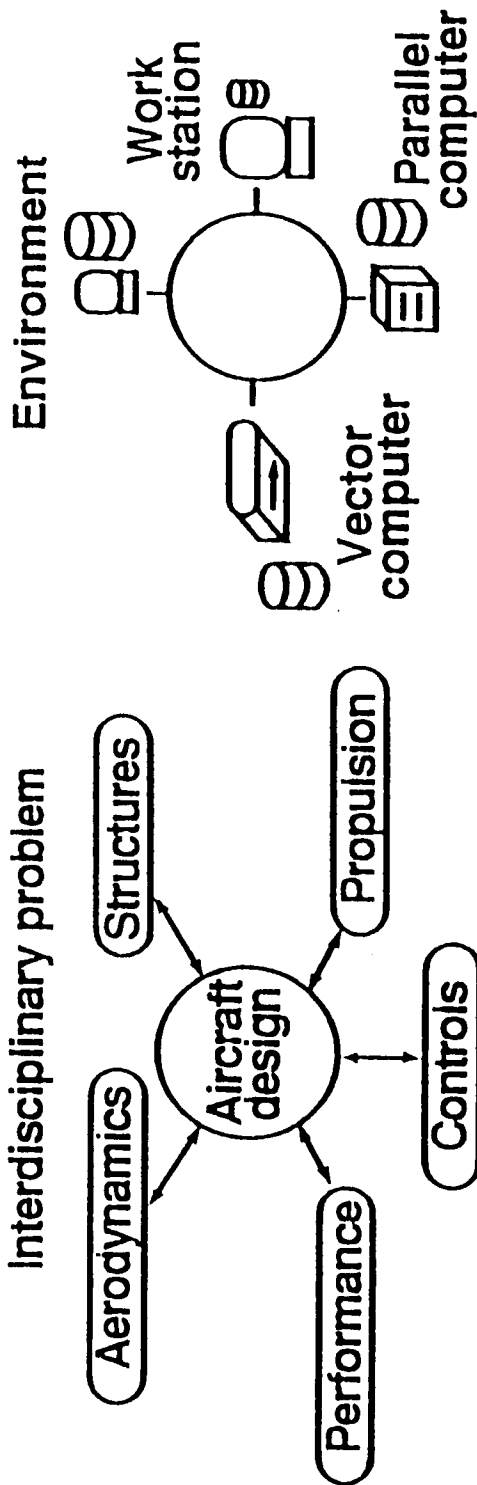
boxes with a 3-D appearance. Doing so brings up an associated window that displays sub-detail for that box. The data display GUI is the third interface element, called SPY, that provides the user with a variety of ways to plot data during the design process. The right part of slide is an example of a color-coded contour plot of wing surface pressures. The buttons at the top of the plot window provide the user with a variety of view controls. In addition to contour plots of aerodynamic pressures and structural stresses on the wing, SPY provides line-plots of cycle history for a variety of design parameters and data results. An example of this for a 20-cycle design iteration is shown in a later slide. This slide illustrates results of using FIDO to minimize the total weight of the HSCT for a 6000-mile range and Mach 2.4 cruise speed. Discipline sensitivity derivatives are calculated using finite differences, and a linearized model based on these is used with the optimization subroutine CONMIN to determine new values for the design variables at each design cycle. The slide illustrates the component weight reduction over a 20-cycle design iteration subject to design constraints on structural stress and deflection. Weights have been non-dimensionalized by nominal payload. The slide also shows the changes to the three aerodynamic design variables (lengths non-dimensionalized by wing span) and two structural design variables (non-dimensionalized by initial skin thickness).



The NASA Langley FIDO project

FIDO - Framework for Interdisciplinary Design Optimization

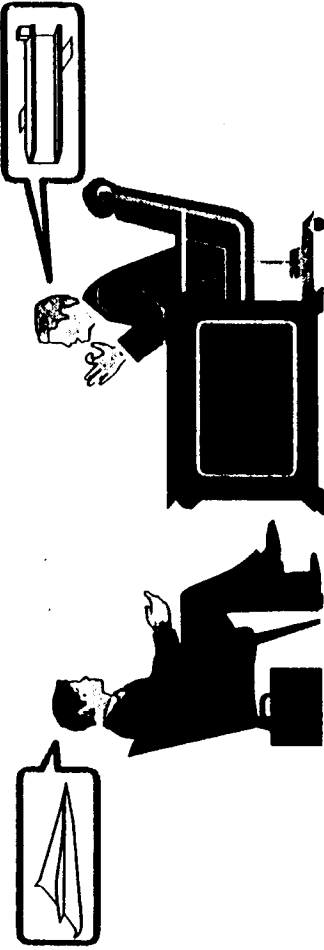
GOAL: Establish a general computational framework, consisting of software tools and programming guidelines, for facilitating the interdisciplinary coupling necessary for multidisciplinary design and optimization problems on heterogeneous computing systems.



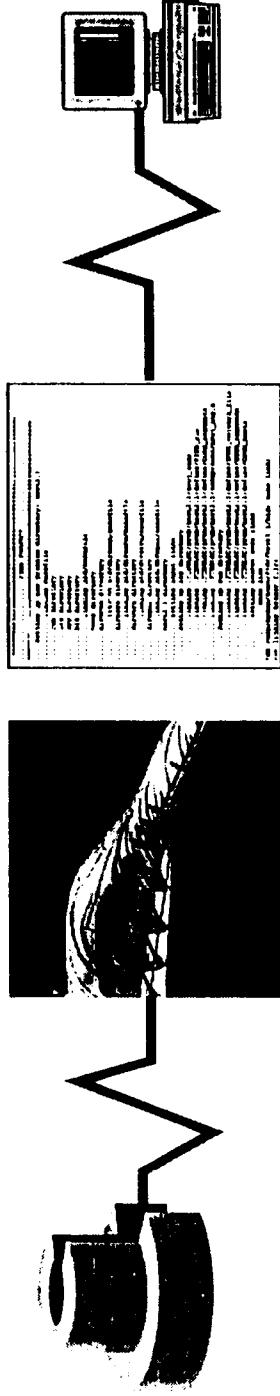
Concept

Develop a Programming Environment for:

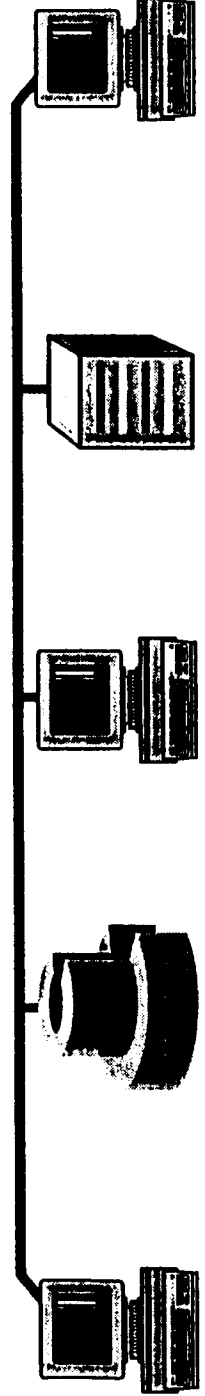
Group of users with diverse specialties



Multiple programs targeted for appropriate computers

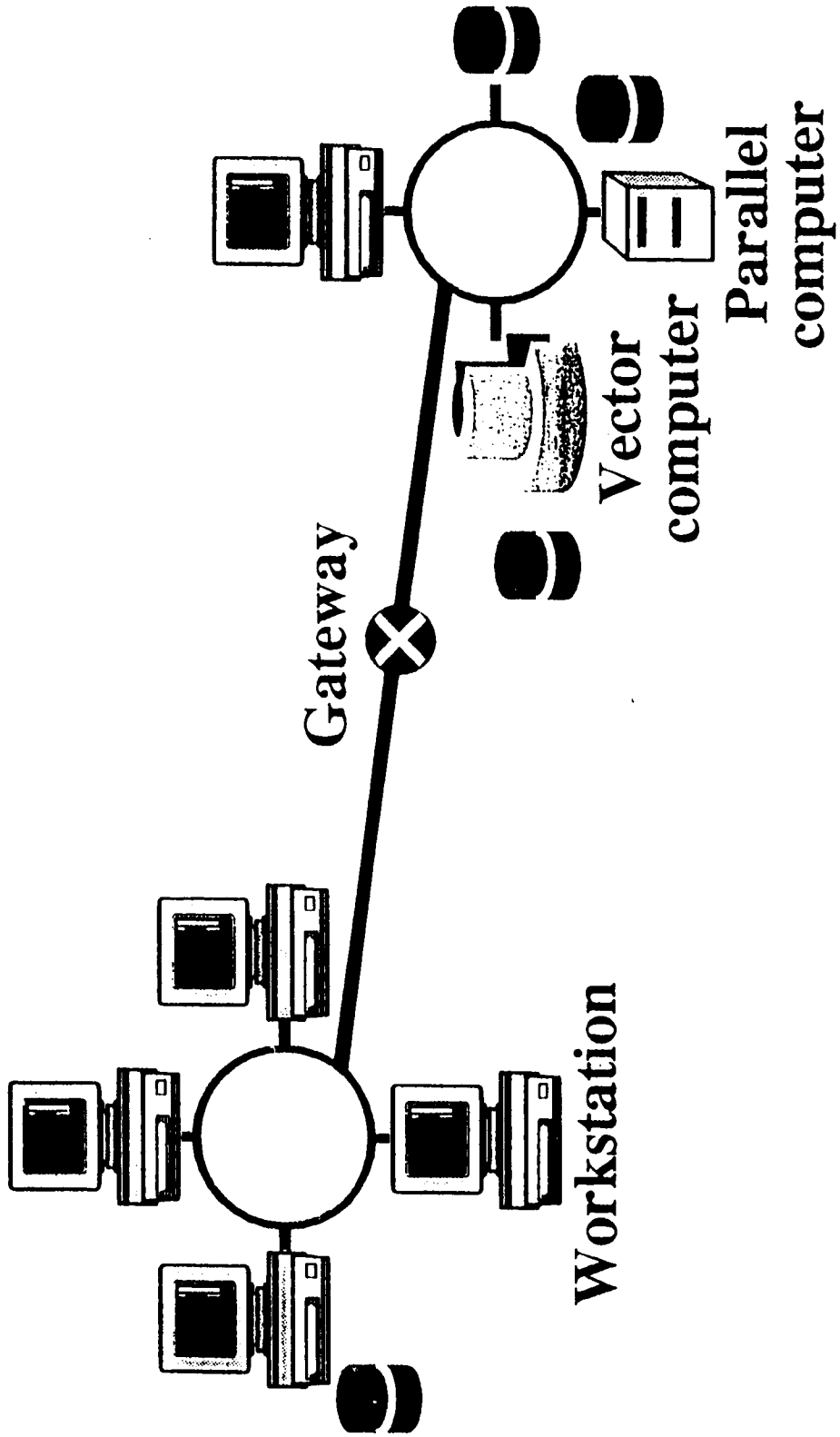


Computers and data distributed over network

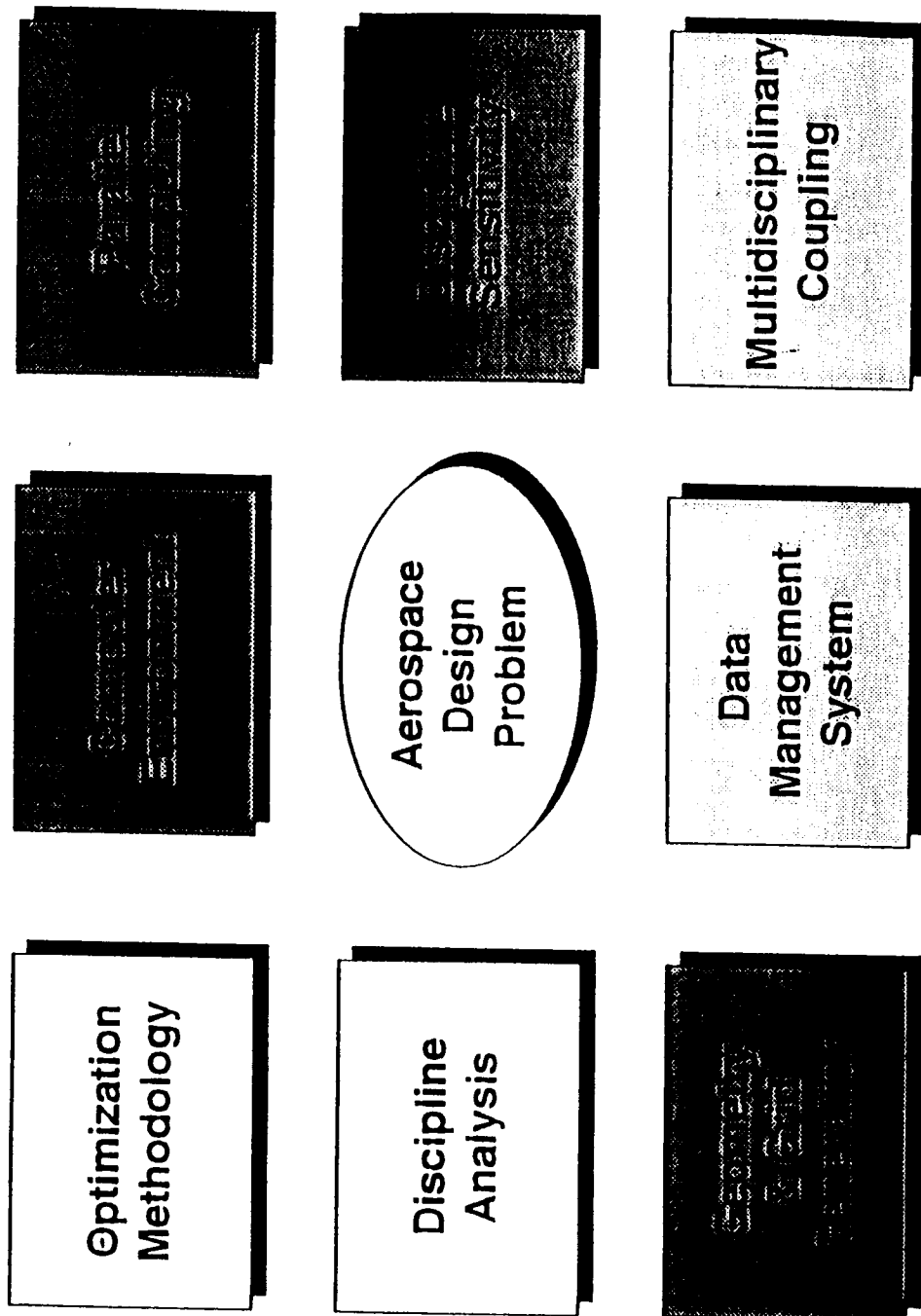


Working together simultaneously on parts of the same problem

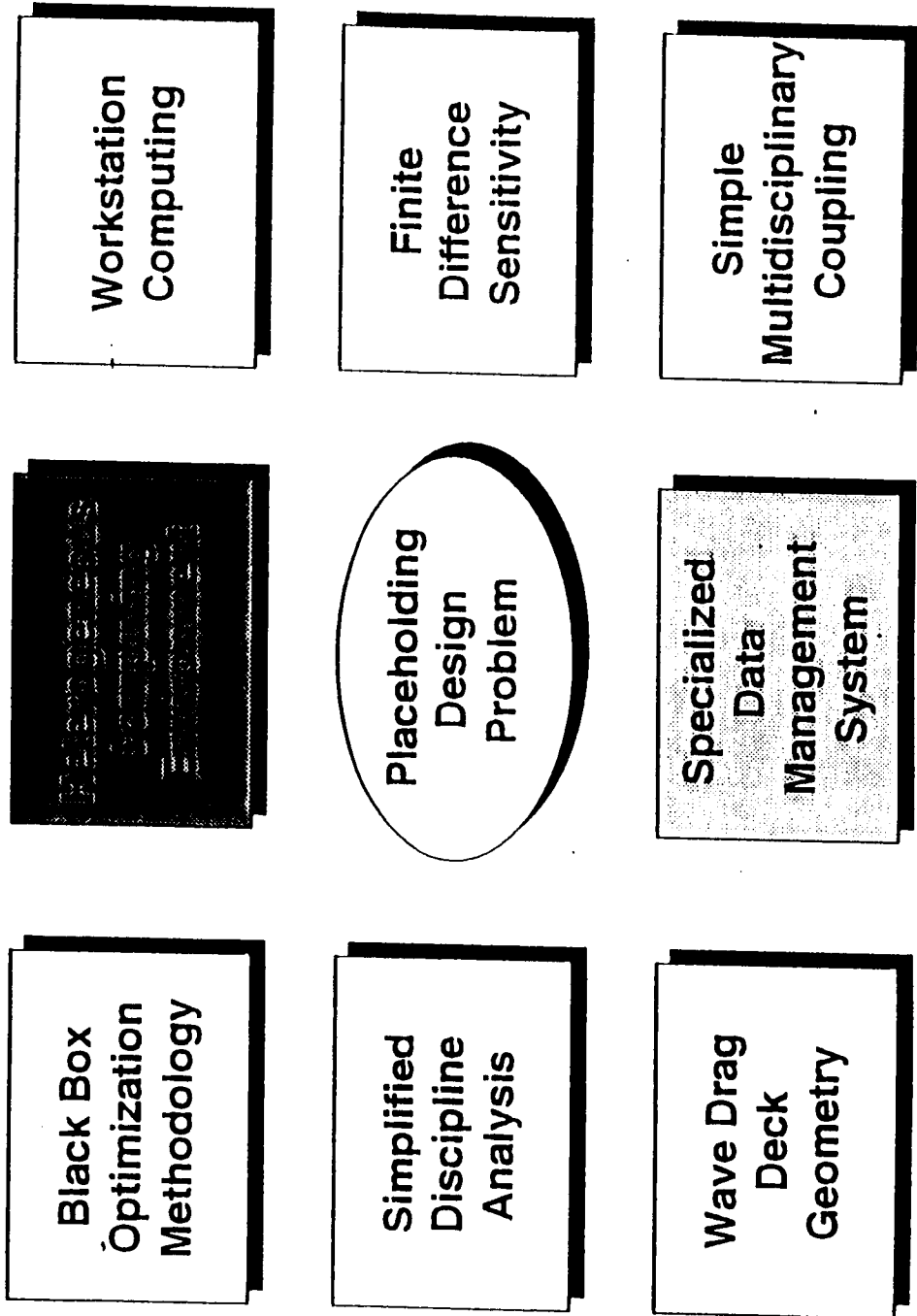
Environment Heterogeneous Distributed Computing



Current LaRC HPCCP Emphasis



Current FIDO Components



ORIGINAL PAGE IS
OF POOR QUALITY

Approach

Limit scope of design/optimization formulation.

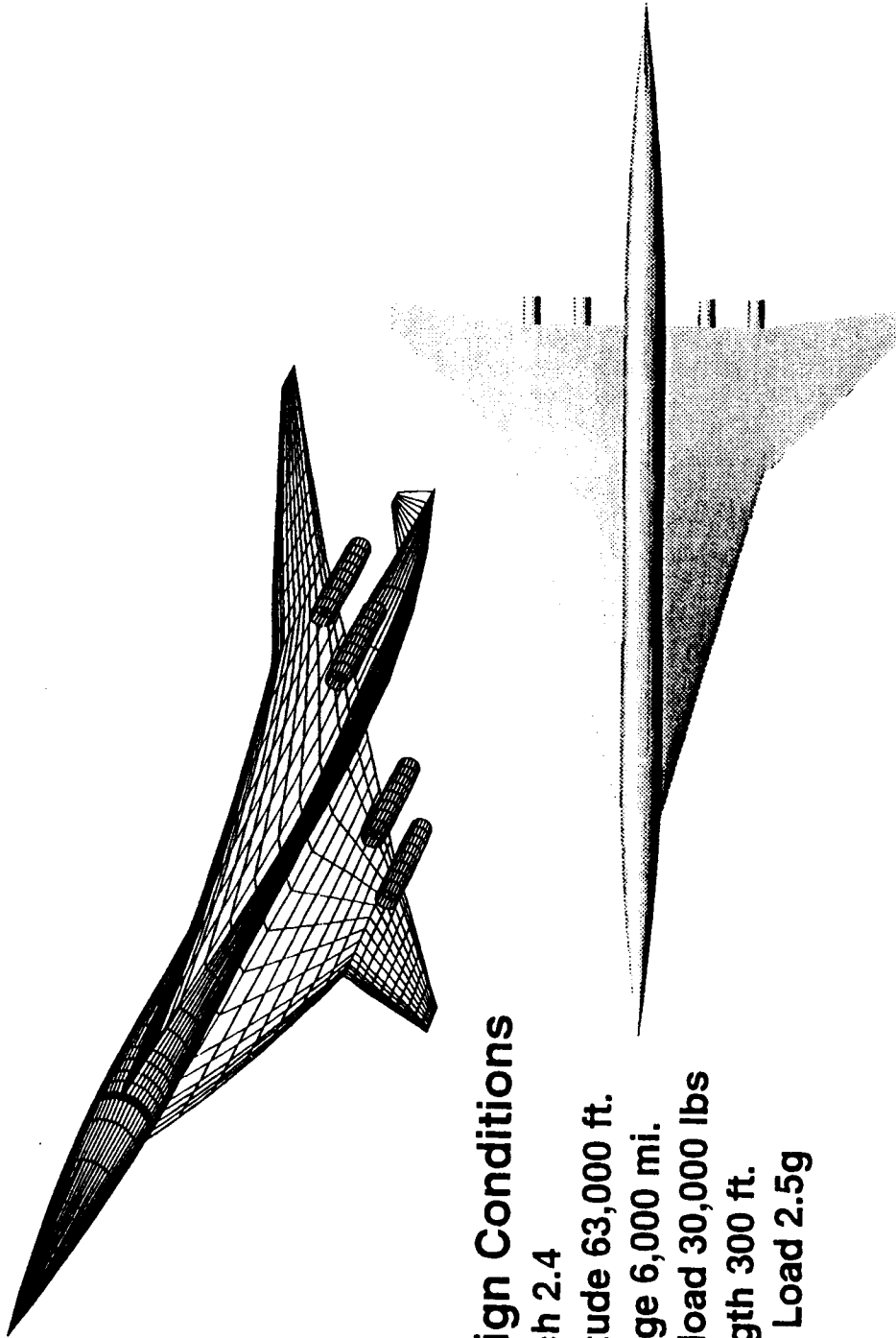
Consider a representative set of disciplines, design criteria, design conditions, design variables, etc.

Use existing engineering analysis codes.

Use a High Speed Civil Transport (HSCT) configuration as the focus problem.

Initial implementation on a distributed workstation environment; migration to parallel testbed in the future

HSTC Baseline Description



Design Conditions

Mach 2.4

Altitude 63,000 ft.

Range 6,000 mi.

Payload 30,000 lbs

Length 300 ft.

Max Load 2.5g

FIDO Project Focus

Short term

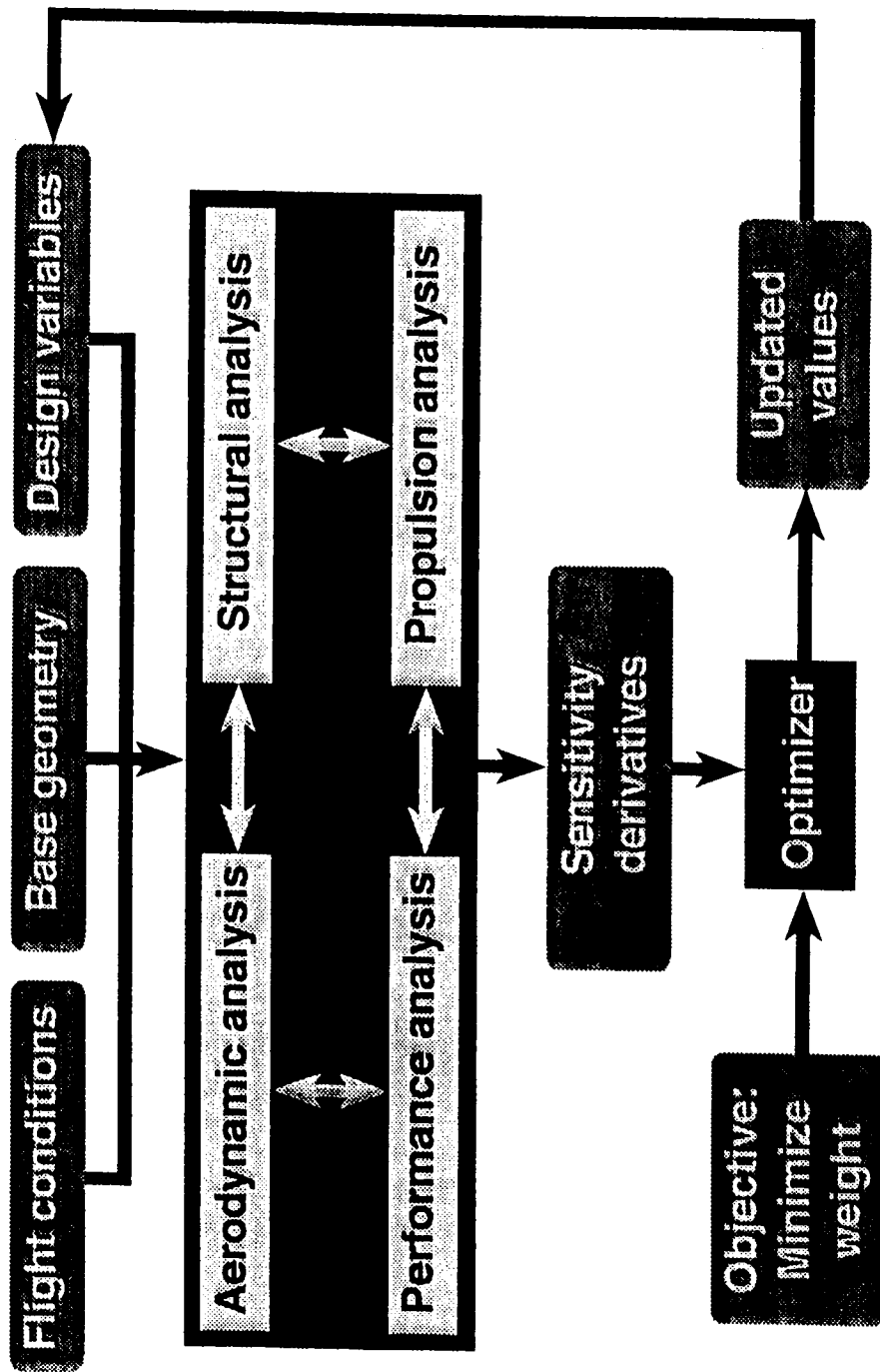
- **simplified model problem**
- **low-fidelity discipline codes**
- **finite-difference sensitivity derivatives**
- **coarse-grain parallelization on a workstation network**

Long Term

- **realistic design and optimization problem**
- **higher fidelity discipline codes**
- **quasi-analytic sensitivity derivatives**
- **coarse-grain parallelization on a heterogeneous computer network along with fine-grain parallelization on massively parallel and vector computers**

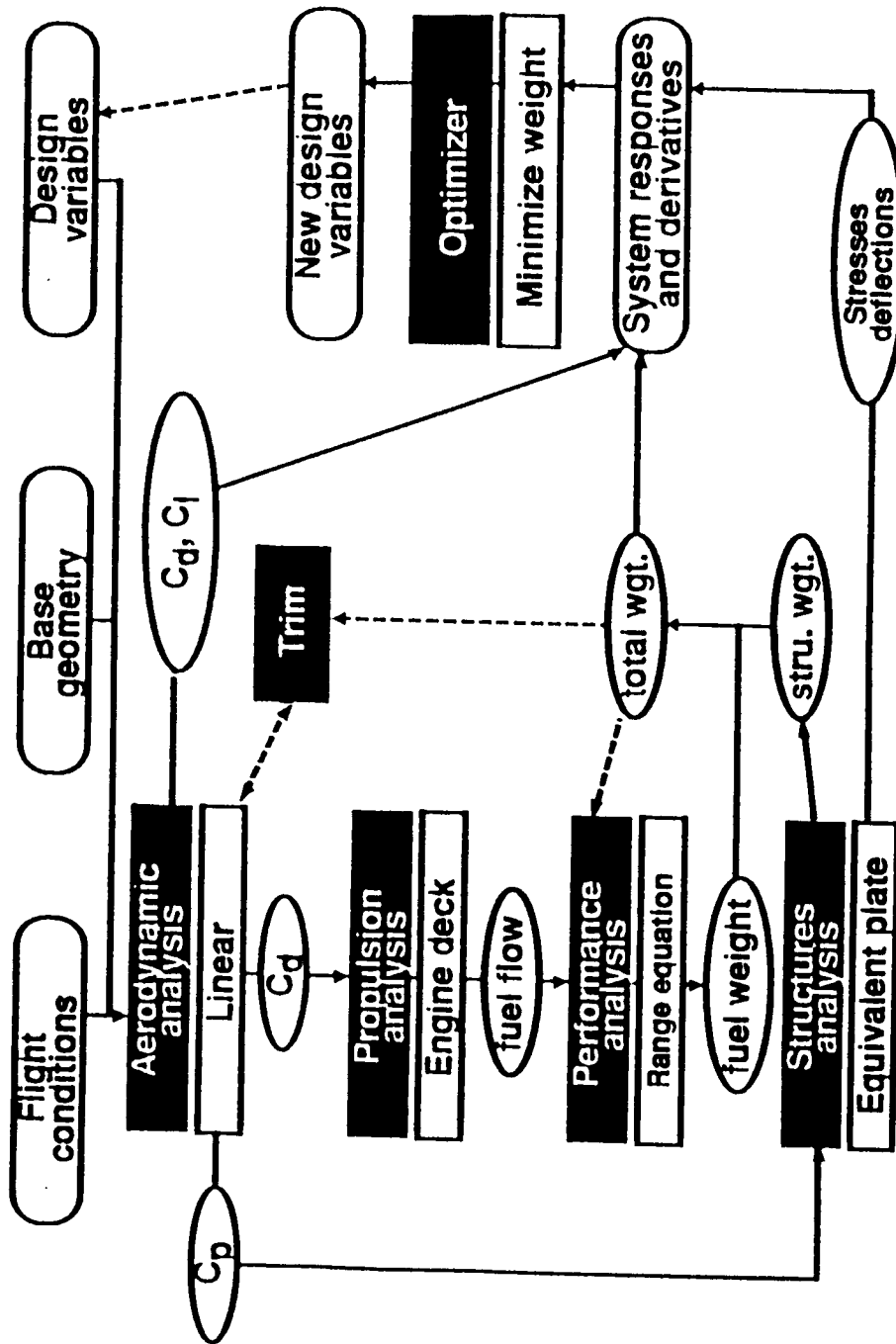
FIDO Example Problem

HSTCT Optimization

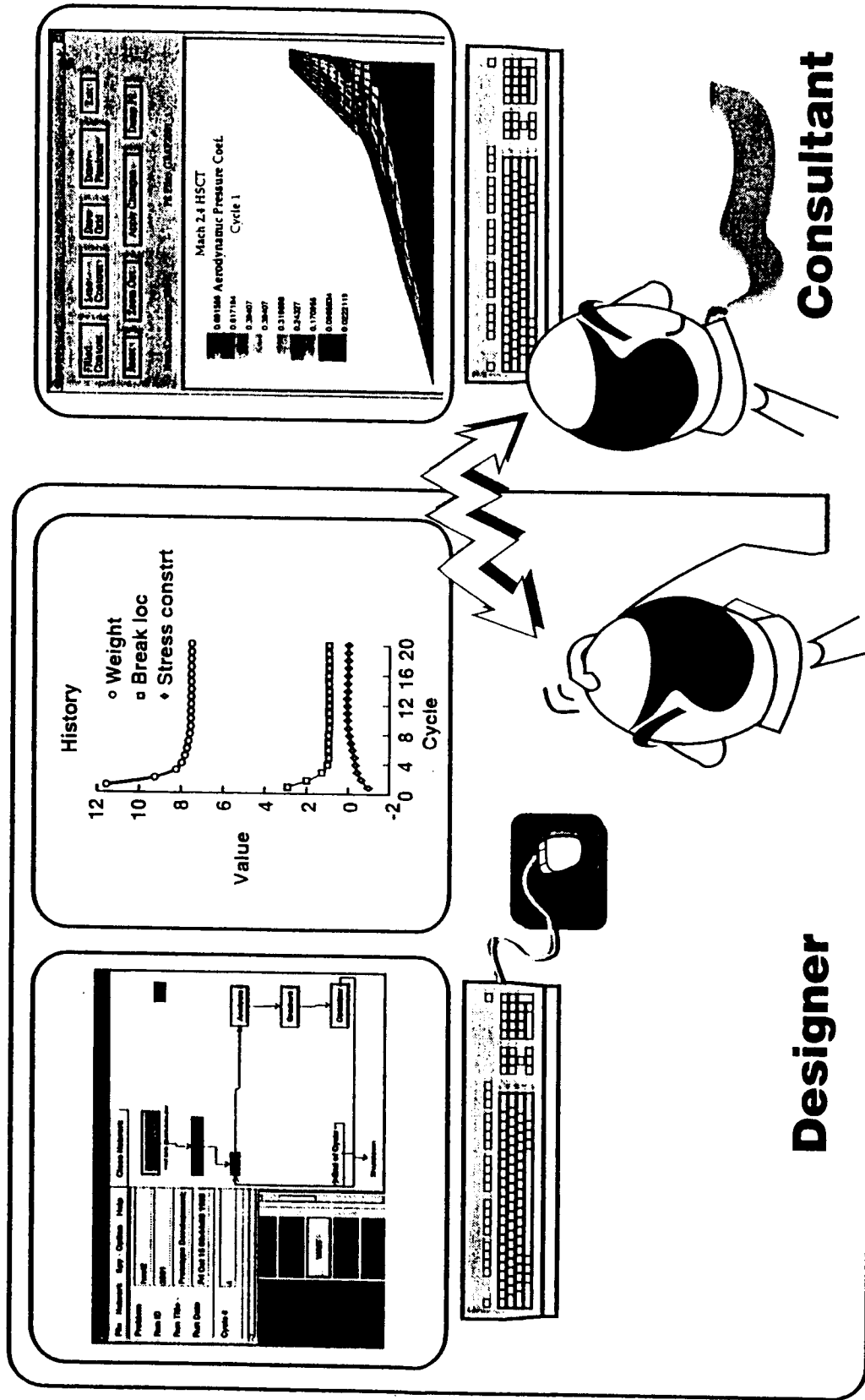




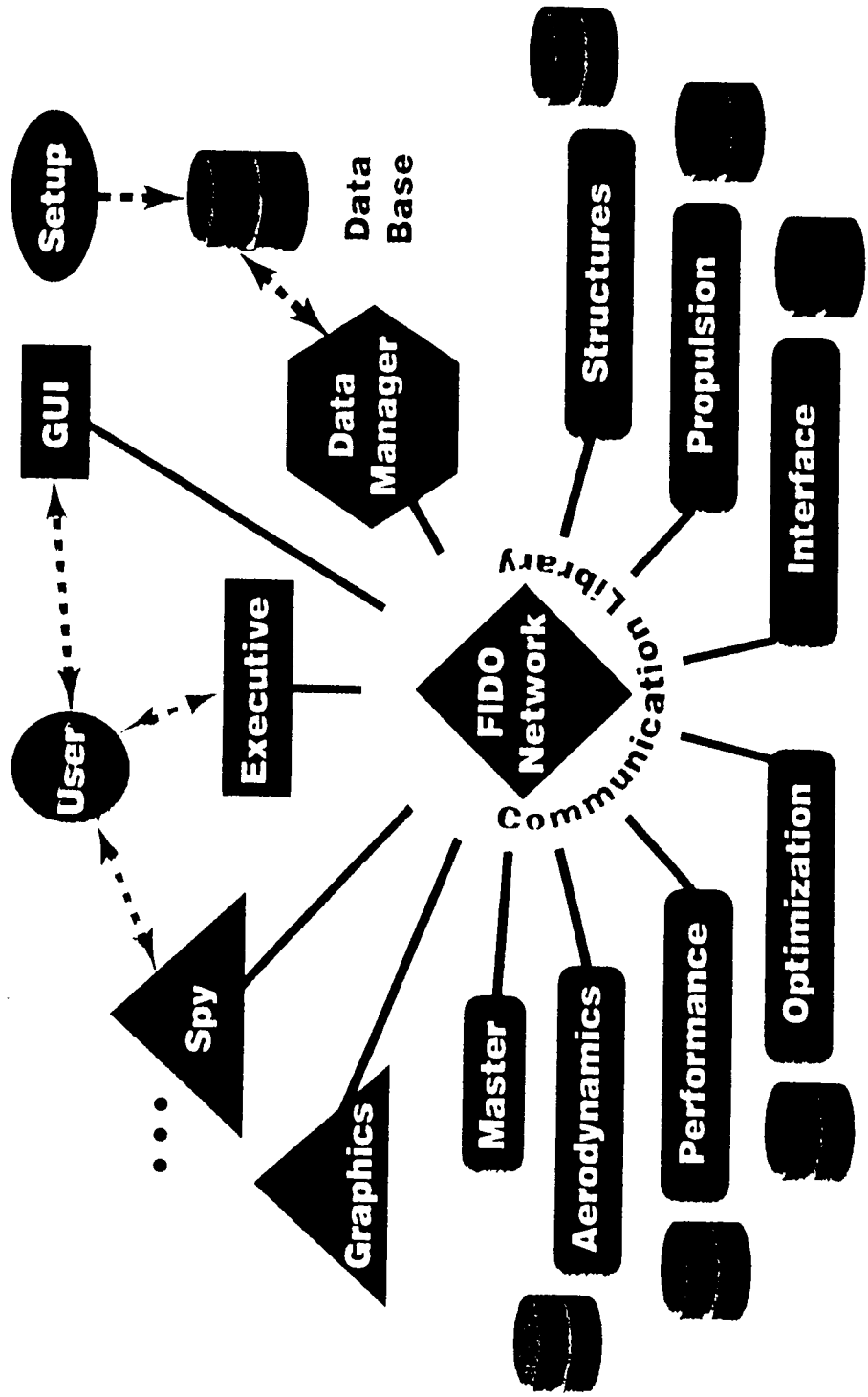
HST problem diagram



FIDO User Interface Concept



FIDO Execution System



Features

- Provides synchronous control in heterogeneous computing environment
- Is adaptable to a wide variety of problems that use multiple computer programs
- Has modular form for easy problem migration
- Incorporates variety of debugging features
- Includes “SPY” capability for -
 - Monitoring computational progress
 - Displaying variables graphically
 - Modifying variables for design steering

FIDO Project Products

Prototype multi-discipline, multi-computer codes

- adaptable
- modular
- portable
- non-proprietary

Prototype toolkit:

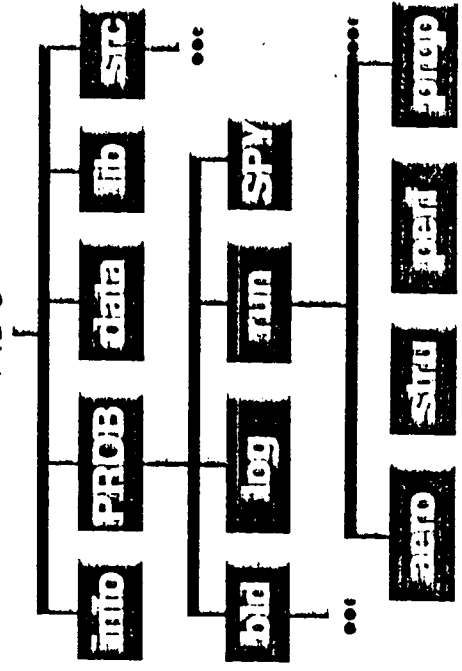
- user interface
- communication library
- data manager library
- driver templates

Software recommendations:

- system software needs
- application programming guidelines

Programming Environment Resources

- Specified code interfaces
- Programming guidelines
 - be adaptable, modular, portable, non-proprietary
- Communications library
 - use message-passing standards
- Database manager
 - use file-based data schema
- Programming templates
- Structured directories FIDO



Low Fidelity Aerodynamics

Based on linear theory

WINGDES : drag due to lift & pressure difference across the wing

AWAVE: wave drag calculation

WAREA/CDFR1: viscous drag (empirical)

Utilizes very simple geometry description.

INPUT

- Geometry Description
- Flight Conditions
- Discretization Parameters

OUTPUT

- Lift Distribution
- Lift and Drag

Low Fidelity Structures

Equivalent Laminated Plate Solution (ELAPS)

ELAPS uses a series of plates defined on wing planform.

Camber and Thickness are defined for each plate.

INPUT

- Forces (acquired from Aero)
- Wing Geometry
- Skin Thickness

OUTPUT

- Structural Weight
- Stresses
- Displacements

Low Fidelity Propulsion

- Level 1 propulsion module from Lewis (TBE_NASA)

INPUT

- Engine Type
- Altitude and Mach No.
- Thrust

OUTPUT

- Fuel Consumption

Low Fidelity Mission Performance

- o Breguet Range Equation

INPUT

- Fixed Range
- Lift and Drag
- Estimated Total Weight

OUTPUT

- Fuel Weight

Approaches to Sensitivity Analysis in CFD

- **Finite Differences**

Compute the sensitivity by taking the difference between two analyses with slightly different values of the design variable.

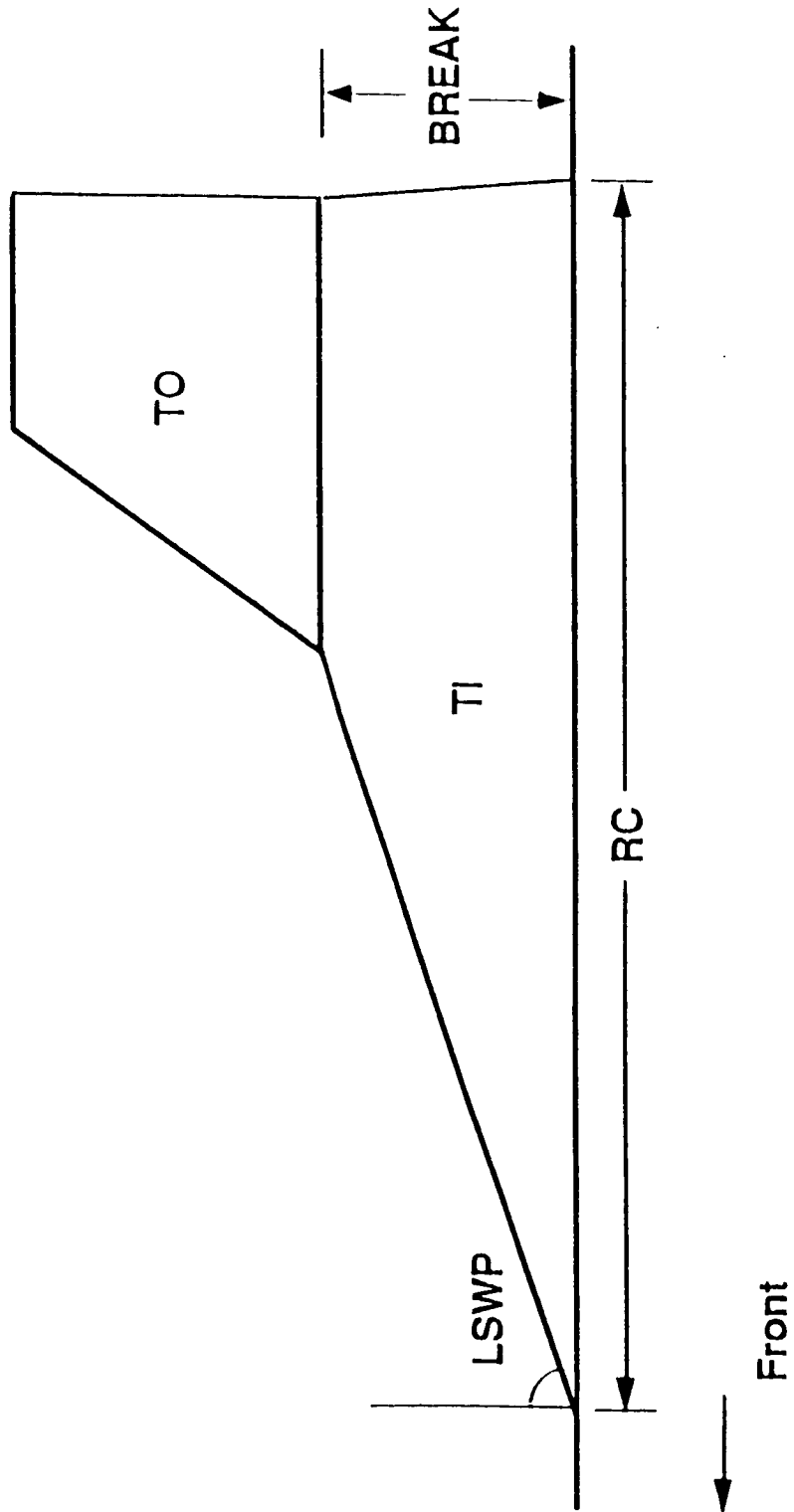
- **Manual Insertion of Quasi-Analytical Sensitivity Derivatives**

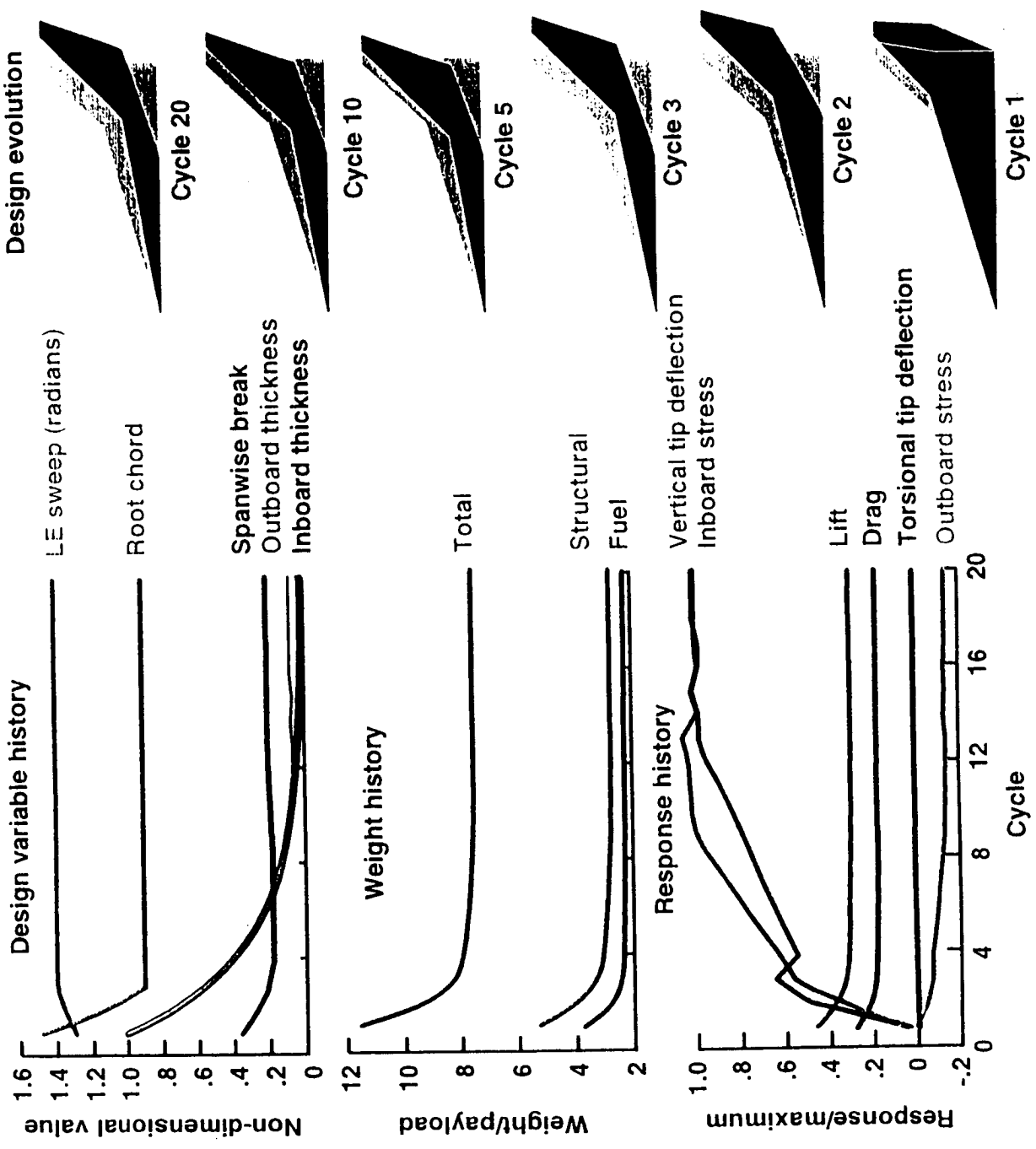
Derive the exact, linear equations satisfies by the sensitivity derivatives and add them to the analysis code.

- **Automatic Insertion of Quasi-Analytical Sensitivity Derivatives**

Utilize a pre-processor which automatically analyzes the Fortran code and adds (automatically) the code for the prescribed sensitivity derivatives.

Wing Planform Design Variables





Accomplishments

The FIDO system has demonstrated -

- Optimization of simple HSC^T design problem
- Use on network of UNIX[®] workstations
(Sun, SGI, DEC)
- Monitoring and graphics display through
“SPY”
- Flexibility of file-based data management

Plans

- ✓ Demonstrate FIDO on a complex HSCCT design problem
- ✓ Replace low-fidelity with high-fidelity analysis computer programs
- ✓ Apply graphical user interfaces for problem setup and for control
- ✓ Fully document the software and publish findings
- ✓ Get and apply feedback from beta tests

FIDO Project Credits

Programming group:

Bob Weston
Tom Eidson
Jim Townsend
Ray Gates
Ramki Krishnan
Ben James
Kelvin Edwards
Bill LaMarsh

... project leader, data manager
... project designer, communications
... aerodynamics, propulsion
... user interface
... aerodynamics
... optimizer
... graphics, system administrator
... structures, performance

Consultants:

Peter Coen
Tom Zang
Gary Giles
Gregg Wrenn
Tom Crockett
Don Randall
Larry Green
Mary Adams

... optimizer, performance
... aerodynamics
... structures
... structures
... communications
... user interface
... aerodynamics, propulsion
... graphics

An Overview of the
Computer Aided Engineering and
Design for Electronics Laboratory

by Shelley Stover

sks@longstreet.larc.nasa.gov

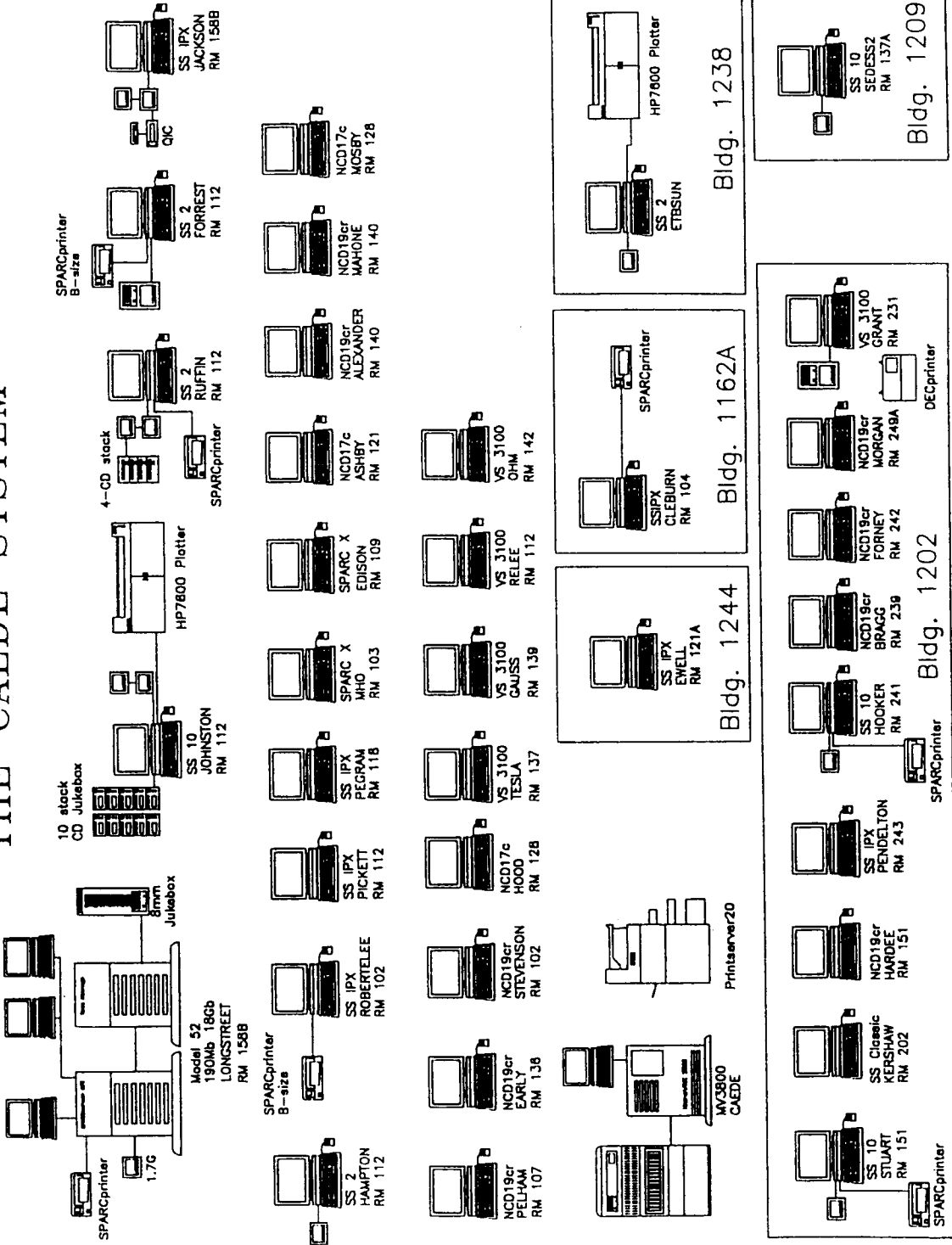
Introduction

CAEDE is a center-wide resource and any Langley employee can obtain a user account.

CAEDE hosts a wide variety of software tools which are applicable to many engineering disciplines such as

- electronic circuit design of programmable integrated circuits and printed circuit boards
- control system modeling
- optical engineering

THE CAEDE SYSTEM



CAEDE Software

Core Tools

Cadence Design Tools

Concept - Design entry
Analog Workbench - Analog design and analysis
Logic Workbench - Digital design and analysis
Leapfrog - VHDL simulator
PIC Designer - PLD design
Profile - Analog HDL simulator
Spice Plus - Analog simulator
Mixed Signal Simulation Environment - mixed board design
OpenSim - Backplane multi-engine simulator
Parametric Plotter - Trade-off and what-if analysis
Smoke Alarm - Stress, Power, dissipation analysis
Veritime - Timing analysis
Allegro - Layout tool

Matlab (Signal Analysis and Modeling)

Matlab - High performance numeric computation
Simulink - Simulates dynamic systems
Signal Toolbox - Signal processing tool
Control Toolbox - Control system engineering
Robust Toolbox - Enhanced control toolbox
Neural Net Toolbox - Neural network tool

Databases

Oracle - Relational database

Documentation

WordPerfect - Word processor

Agency - Wide:

IHS ICSC - IC/Discrete Parameter database
IHS Recal/Z - Passive component database
NAS - NASA Assurance System

Hosted Tools

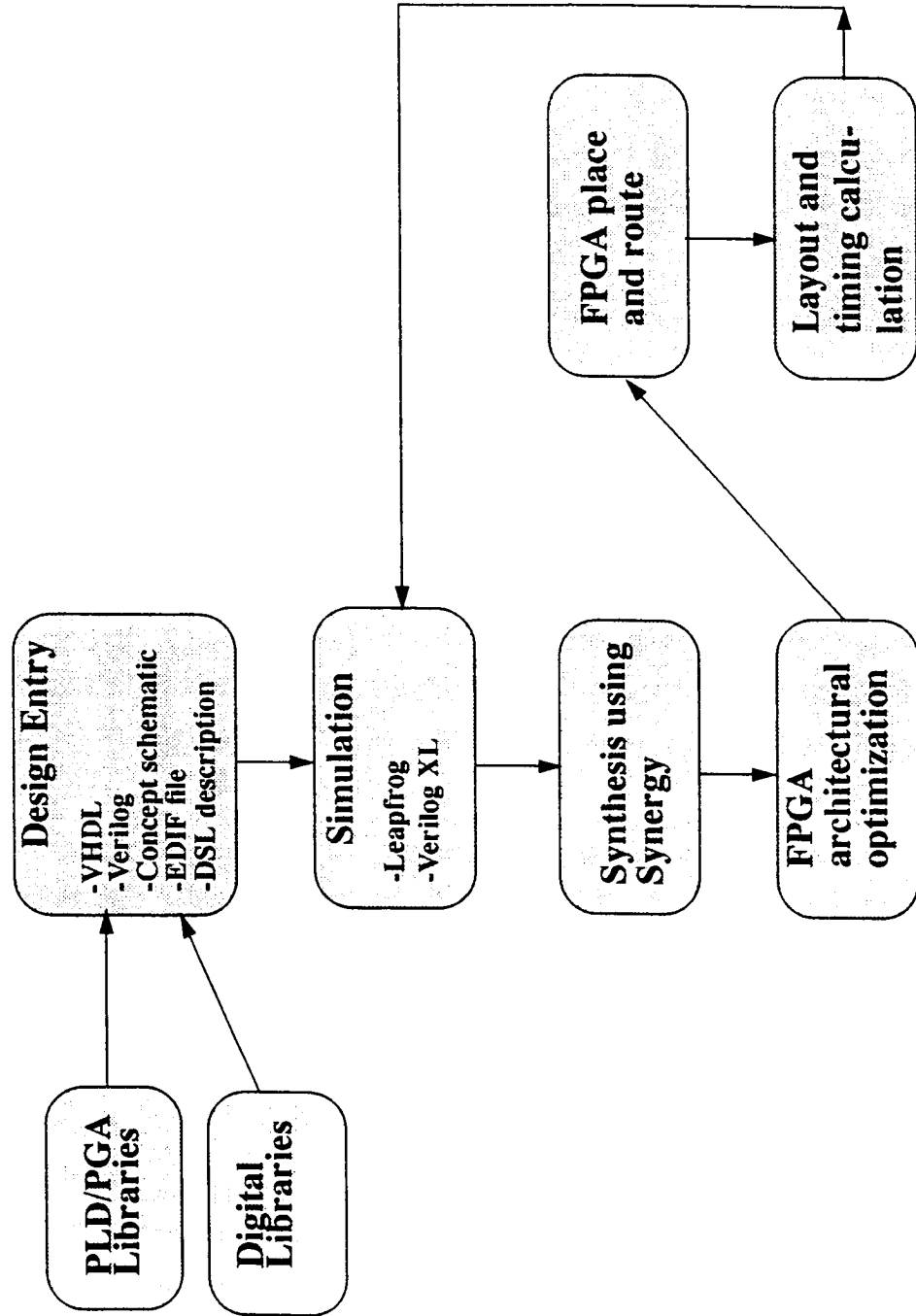
Verilog XL - Verilog simulator
Vanguard - Design entry
Composer - Design entry
Silvaco - 2D device simulator framework
PSpice - Digital and analog circuit simulator
Labview - Virtual instrument workbench
CodeV - Optical design and analysis
Autocad - CAD package
CorelDraw - drawing package
Framemaker - Word processor
TMS320C compiler and simulator
Softwindows- Microsoft windows emulator
C & C++
ADAS - Architectural design and assessment
Foundry libraries

The Cadence Design Tools

CAEDE hosts the following Cadence tools for the design of integrated circuits and printed circuit boards.

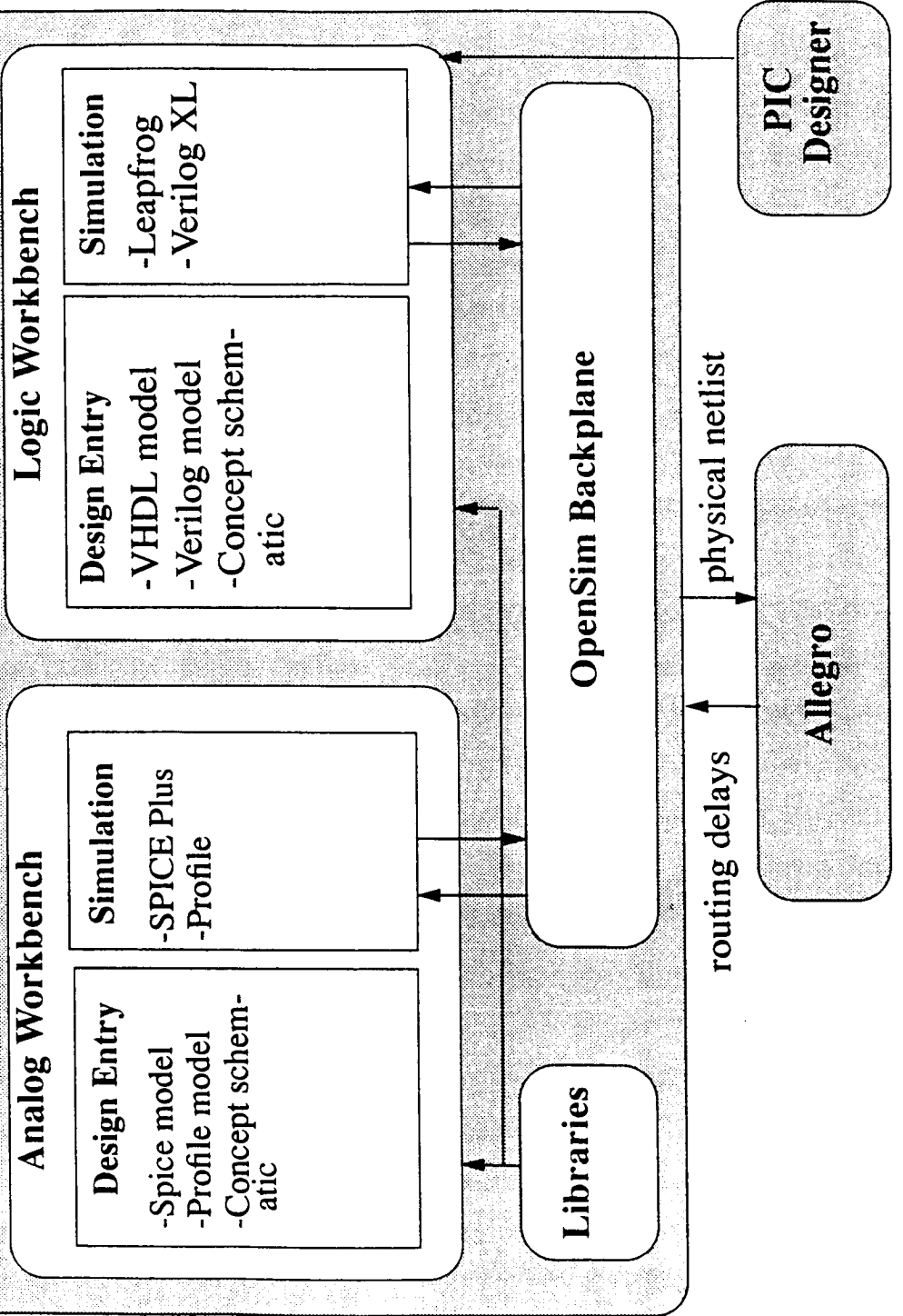
- Concept - schematic design entry
- Composer - schematic design entry
- Verilog XL - Verilog compiler and simulator
- Leapfrog - VHDL compiler and simulator
- Profile - analog design language compiler and simulator
- PIC Designer - programmable integrated circuit design environment
- Logic Workbench Designer - digital design environment
- Analog Workbench - analog design environment
- Mixed Signal Simulation Environment
- Allegro - layout of printed circuit boards

The Cadence Integrated Circuit Design System: PIC Designer



The Cadence PC Board Design System

Mixed Signal Simulation Environment

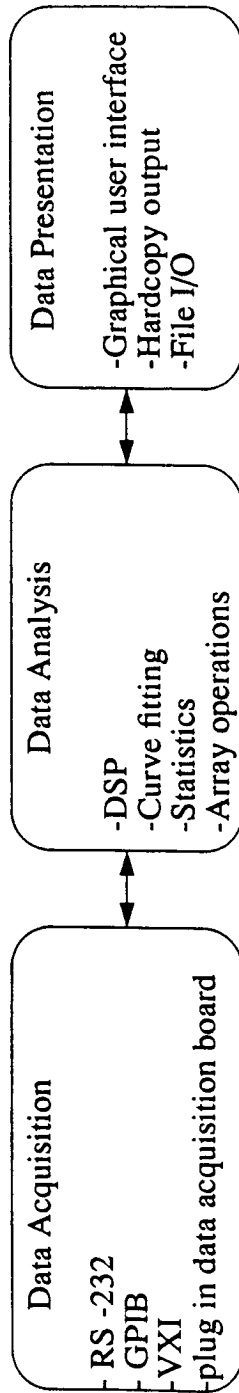


Electronics Parts Database

- CAEDE hosts an on-line database of electronic part databooks and specifications which is updated bi-monthly.
- The database contains over 2 million integrated circuits, discrete semiconductors, and passive components. The parts can be searched by part number, characteristics, and function.
- The software is from Information Handling Systems (IHS) and is contained on 60 CDs in 10 jukeboxes. Our license is agency wide and funded by Code QE.

Labview

Data flow in Labview

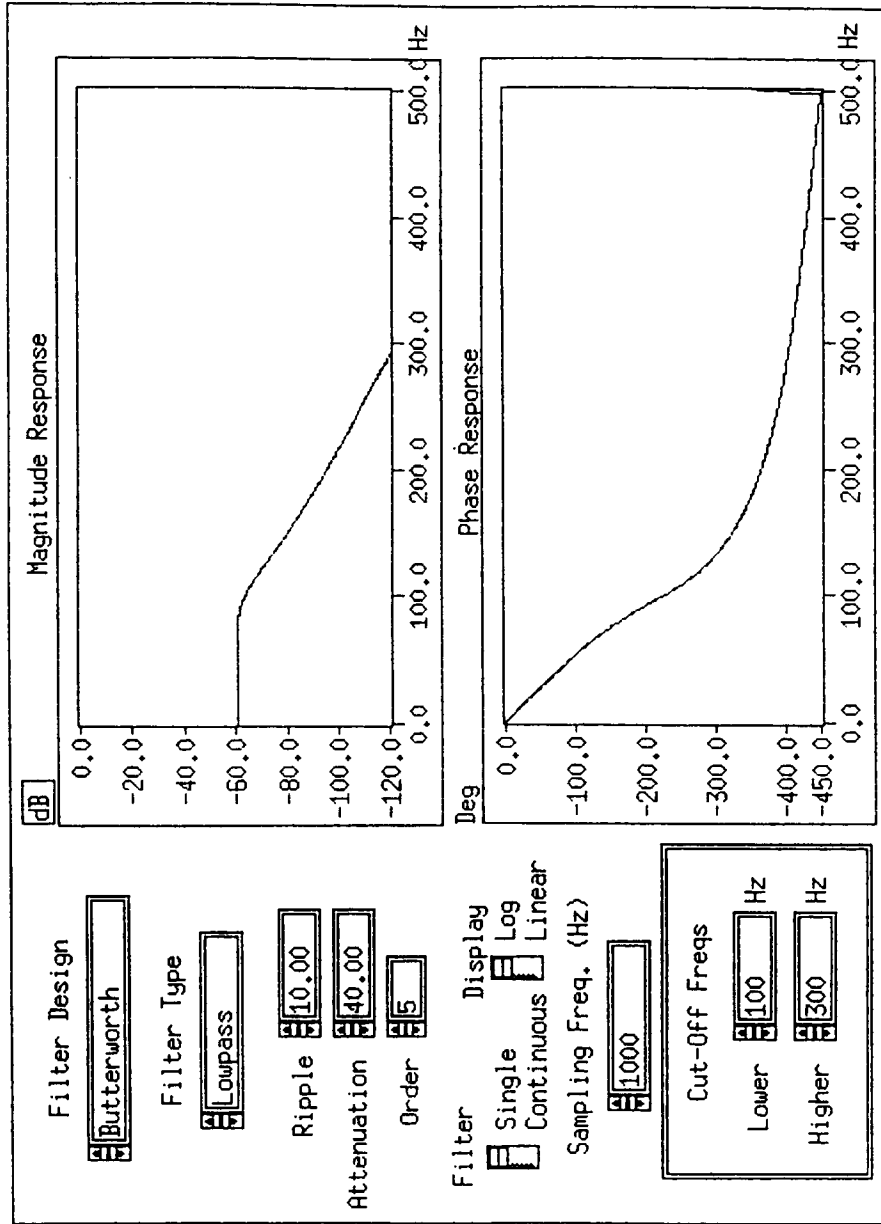


Labview Virtual Instrument Components:

- Front panel - interactive interface to supply inputs and observe outputs of the instrument
- Top-level block diagram - used to program the VI for acquisition, analysis, and formatted I/O
- Low-level panels and diagrams - together these components form sub VIs which can be used as instruments in other VIs

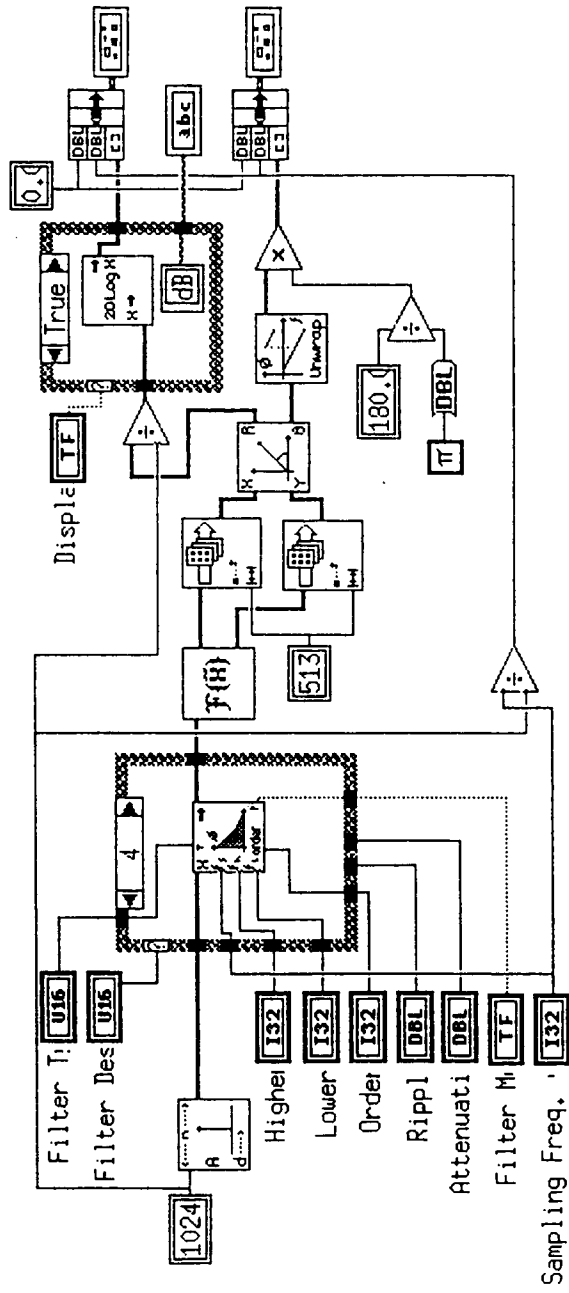
Front Panel of IIR Filter Example

Front Panel



Block Diagram of IIR Filter Example

Block Diagram

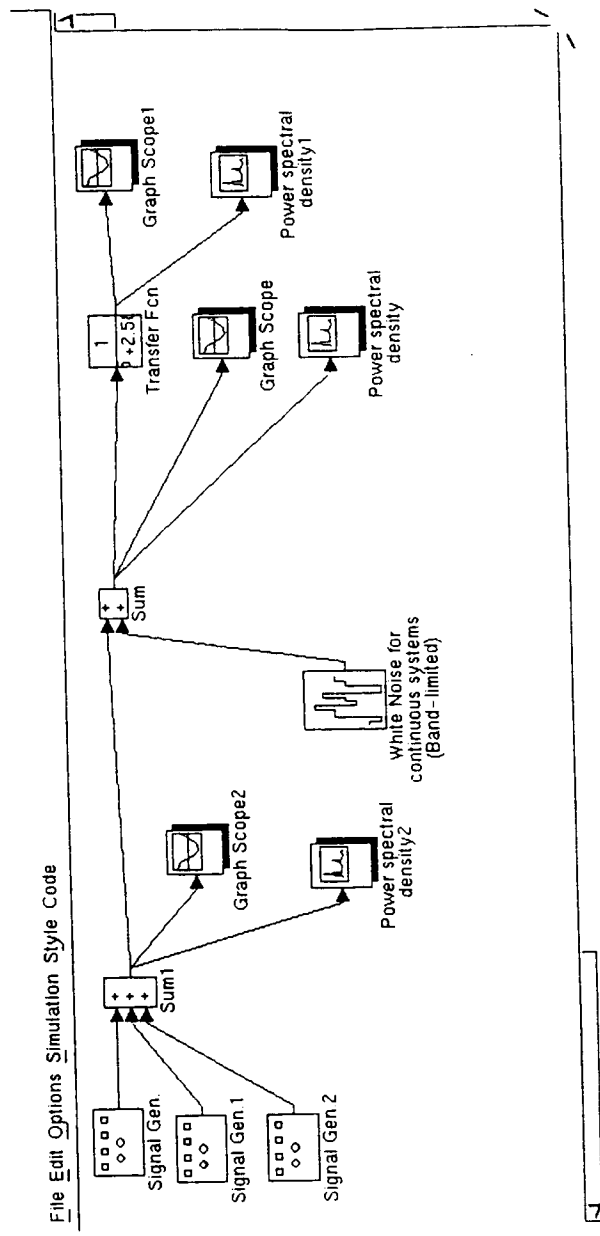


MATLAB

Matlab

- Matlab is a high performance package for engineering and scientific computations. It allows users to manipulate matrices in a fraction of the time that it would take to write C programs, etc. Also, Matlab is able to generate several different 2D and 3D graphs.
- Matlab toolboxes:
 - Control System Toolbox
 - Signal Processing Toolbox
 - Robust Control Toolbox
 - Neural Network Toolbox
- SIMULINK - Provides a graphical user interface for modeling and simulation of dynamic systems.

Simulink Block Diagram



Optical Research Associates - Code V

- Code V is used for the design, optimization, tolerancing, costing, manufacturing, and alignment of optical systems.
- One of its primary uses is the development of optical systems from concept through final design.
- It handles many types of surfaces including aspherics, toroids, gratings, diffractive optical elements, and many others.
- Code V can also be used for cost estimates, environmental analysis, image simulation, and it interfaces with NASTRAN, interferometers, and CAD/CAM packages via IGES.
- Code V was used for the design and analysis of the LASE and LITE optical systems.

Silvaco

- Composed of the S-Pisces, Blaze, and Luminous simulators and the graphics package Tonyplot
- S-Pisces is the two-dimensional device modeling program that simulates the electrical characteristics of common MOS and bipolar integrated circuit elements
- Blaze contains material libraries for exotic semiconductors such as SiC and GaN as well as the codes for simulation of heterostructure devices.
- Luminous implements models for optical generation of carriers and allows simulation of a broad range of devices .

Training

- The following Cadence training videos are available:

- Concept
- PIC Designer with Concept
- Top-Down Design with Logic Workbench
- Designing with Leapfrog
- Introduction to VHDL
- Veritime
- Allegro Automatic
- Allegro Interactive
- Analog Workbench II with Concept
- Analog Workbench Mixed-Signal

NOTE: These videos are accompanied by training manuals, lab manuals, and lab software and are updated as new versions are installed.

- On-line tutorials are available for Matlab, Labview and Simulink
- Basic UNIX short courses are provided by CAEDE system administrators

General Information

- CAEDE accounts are available to all center employees. To request more information or obtain a user account form contact one of CAEDE's system administrators:

Steve Comer: steve@longstreet.larc.nasa.gov, extension 43710

Don Brandt: dtb@jackson.karc.nasa.gov, extension 43716

- For general information on CAEDE and the Information Handling Services see the MOSAIC CAEDE Home Page.

- For IHS technical information contact

Randy Regan: crr@longstreet.larc.nasa.gov, extension 41869

The Software Engineering and/or Ada Lab (SEAL)

Presented at the
"Role of Computers in LaRC R&D" Workshop
June 16, 1994

Robert Kudlinski
Information Systems Division

Software Engineering and/or Ada Lab (SEAL)

BACKGROUND (and HOME PAGE)

- 1989: ACD (now ISD) was charged to manage, develop, and assure mission critical software systems for several on-going and all new LaRC space-flight projects
- 1990: The SEAL was a new-start to meet this requirement by implementing a common software engineering process (people, procedures, tools) across these projects
- 1992: Selected to participate in the NASA Software Engineering Program
- Since inception, the SEAL has received increasing requests from other LaRC software organizations/domains seeking to improve their processes. Existing and future plans to help transfer software engineering technologies are presented.

Software Engineering and/or Ada Lab (SEAL)

LaRC SPACE-FLIGHT PROJECTS

- The SEAL has supported numerous projects at various lifecycle phases: CERES, CSI, DGV, JADE, LITE, MIDAS, RAME, ROVER, SABRE, SAFIRE, SAGE III, SEDS, SUNLITE, and TRACER
 - » Primarily remote sensing, active control of space structures and technology demonstration experiments
 - » On-board, embedded flight computers (primarily 80x86) for real-time instrument control and data acquisition (3,000 - 25,000 SLOC)
 - » GSE computers (primarily 80x86 PC's) for instrument development, test, calibration and mission operations (5,000 - 70,000 SLOC each)
 - » Missions from 3 days to 5 years, development schedules of 2-5 years
- Many technical and management challenges:
 - » Real time, embedded, and non-deterministic systems require specialized tools and practices
 - » Physical constraints (power, weight, radiation) severely limit computer resources (CPU speed, memory, I/O) and demand optimization
 - » Insufficiently defined, continually expanding requirements
 - » Trading off quality and reliability with tight manpower and schedules
 - » Short term vision of projects does not help process improvement

Software Engineering and/or Ada Lab (SEAL)

HUMAN RESOURCES

- Education and training
 - » Offering 10-15 classes per year through Training Office such as Ada, object oriented design, CM, real-time programming, rate monotonic analysis and system engineering
 - » Developing, documenting, and video taping specialized in-house training such as Ada programming, formal inspections, and using real-time, embedded systems tools
- Information Resources
 - » Library (books, periodicals, standards, guidebooks)
 - » Electronic information exchange and communications network
 - » Providing ISD user consultation service for Ada and software engineering questions
- Projects were assigned and the SEAL became a new start one year before the current climate of decreasing civil service and NPS, which has curtailed planned activities

Software Engineering and/or Ada Lab (SEAL)

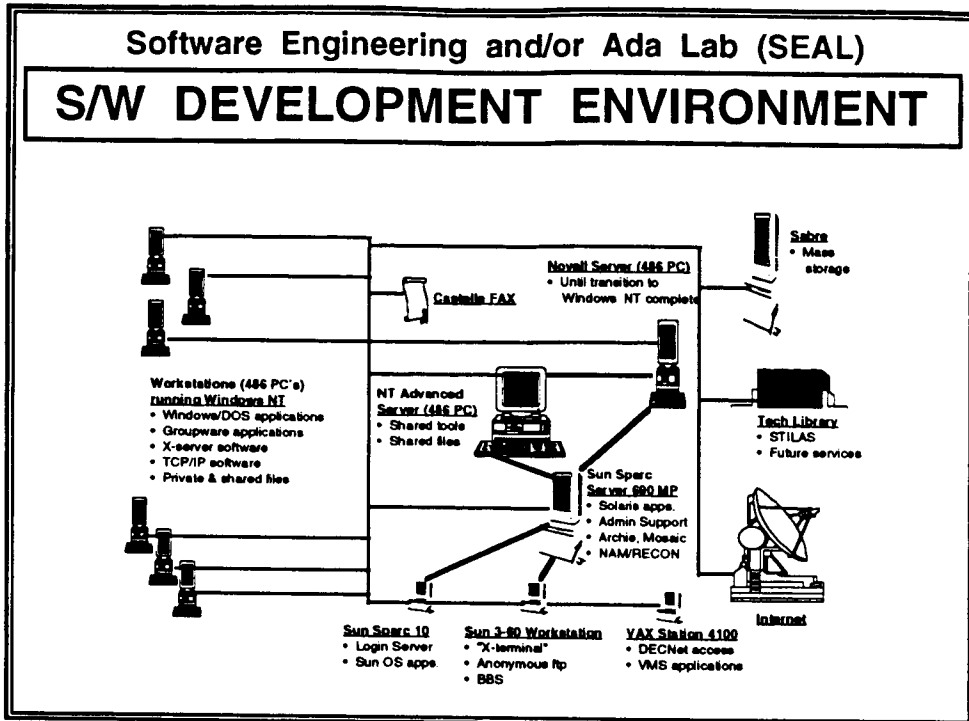
PROCEDURE GOALS

- **Standardize on and reuse common procedures, expertise, tools, and products across projects**
 - NASA and other software standards as available and appropriate
 - Object-oriented requirements analysis and design methods
 - Ada used as the primary programming language, some C, C++
- **Define and document key, repeatable software development procedures as baseline for future process improvement and training new employees**
 - Formal inspections guidebook completed
 - Configuration Management Guidebook in review process
 - Guidelines for selected real-time, embedded system tools completed
 - Evaluating existing, documented IV&V procedures
 - Other procedures under development
- **Moving toward a complete software lifecycle approach based on evolutionary spiral model**
- **Contact Pat Schuler at 4-6732 for more information on SEAL procedures**

Software Engineering and/or Ada Lab (SEAL)

TOOLS

- **Operating a distributed software development environment via LaRCNET**
 - Compilers, CASE, CM, and project management tools in place
 - Beginning to use InQuisix Reuse tool
 - Reverse engineering and code analysis tools
 - Electronic information management and communication tools
 - Plan to increase use of automated code generators and testing tools
 - Work from desk, SEAL, hardware development labs and test facilities
 - Standardized environment allows sharing of tools to reduce project cost and effectively shift personnel in response to changing priorities
- **Real-time, embedded analysis tools**
 - Embedded system cross compilers
 - Emulators and logic analyzers for 80x86 and 1750A processors
 - Functional equivalent 80x86 and 1750A flight computers
- **Contact Jerry Garcia at 4-5888 for more information on SEAL tools**



- Software Engineering and/or Ada Lab (SEAL)**
- NASA S/W ENGINEERING PROGRAM**
- One Code QE mission has been to improve the quality and reliability of software products developed for space flight projects, both manned and unmanned, at the NASA Centers
 - The NASA Software Engineering Program was initiated in 1991 to establish and grow Centers of Excellence across the Agency:
 - JSC: Shuttle Data Systems Branch
 - GSFC: Flight Dynamics Division (Software Engineering Lab)
 - LaRC: Flight Software and Graphics Branch (SEAL)
 - Long term vision of the Program is to put self-sufficient, continually improving processes in place, establish standards, and then use these organizations to transfer effective technologies to other NASA domains
 - LaRC tasks have been primarily in the areas of:
 - Capturing and documenting the SEAL software development process for small Ada projects and assessing Ada's impact
 - Improving software technology transfer methods and software reuse
 - Evaluating IV&V procedures on LaRC projects and research testbeds

Software Engineering and/or Ada Lab (SEAL)

EXISTING GENERAL SUPPORT

- Coordinate 10-15 widely attended software engineering courses per year and sponsor educational presentations (e.g., LaRC/NASA management, LCUC, Local Universities, Conferences)
- Implement and fund formal inspections program widely used across LaRC (training, pilot projects, implementation guidebook)
- Serve on numerous review panels, technical committees and QAT's
- Transferring software technologies to other domains, such as NTF DAS, flight simulation, HSR, and TAP programs - currently manpower limited
- Open access to SEAL tools via LaRCNET - currently manpower limited for training and consulting on these tools
- Access to real-time, embedded system tools possible, but requires proper training and procedures to be followed
- Open access to library (books, periodicals, standards, guidebooks)
- Started the Software Quality, Productivity and Reliability Team to promote communication and coordination of software engineering efforts at LaRC

Software Engineering and/or Ada Lab (SEAL)

FUTURE GENERAL SUPPORT

- Continue existing work on previous chart, expanding where possible:
 - » Proposal accepted by Code Q that 90% of SEAL funding (FY 95-98) specially targeted for manpower to transfer software engineering technologies to on-going LaRC research programs
 - » Decreasing project load should provide ISD opportunity to shift resources from space-flight projects to general support
 - » Partnerships are welcomed
- Increase use of electronic information dissemination, particularly MOSAIC

SESSION 9 CAE Tools

Chaired by

Carol D. Wieseman

- 9.1 Digital Control of Wind-Tunnel Models Using LabVIEW - Sherwood T. Hoadley
- 9.2 Electronic Engineering Notebook: A Software Environment for Research Execution, Documentation, and Dissemination - Dan Moerder
- 9.3 IDEAS² Computer Aided Engineering Software - Pat Troutman
- 9.4 Matlab as a Robust Control Design Tool - Irene Gregory
- 9.5 Simulation of the Coupled Multi-Spacecraft Control Testbed at The Marshall Space Flight Center - Dave Ghosh, and Raymond C. Montgomery

Digital Control of Wind-Tunnel Models Using LabVIEW

by
Sherwood T. Hoadley

presented at the
LaRC Computer Systems Technical Committee Workshop on
The Role of Computers in LaRC R&D
June 15-16, 1994

Digital controller and data acquisition and analysis systems were developed for several wind-tunnel models which use National Instruments LabVIEW[®] Software and National Instruments Hardware within a Macintosh environment. The objective of this presentation is to illustrate the use of LabVIEW for interactive animated display of acquired experimental data and real-time control of some wind-tunnel models.

The first system illustrates a flutter suppression system (FSS) which was used to suppress flutter for a small piezoelectrically actuated wing in a small flutter research and experiment device (FRED) with a 6"×6" test section. The following illustrations are included which show various aspects of the FSS system:

- A photo of FRED and a flow diagram of the wind tunnel
- A block diagram of the closed-loop system
- The digital control system software schematic of the LabVIEW user interface routines on the Macintosh and the real time system comprised of boards plugged into the Macintosh Nu-bus but sharing their own real-time RTSI bus.
- The front panel of the FSS LabVIEW Controller virtual instrument (VI) interface to the real-time controller digital signal processor (DSP)
- Results of open and closed loop strain response to wind-tunnel turbulence

The next LabVIEW VI which is illustrated is an instrument which interfaces with a data logger which samples various thermocouples and sends the requested data to the Macintosh for display and storage. Two figures included are:

- The front panel depicting a beam clamped to a table with thermocouples placed at various locations and a strip chart displaying the data.
- The block diagram of the code for the data logger which shows the way in which LabVIEW is coded. As indicated, the code is a flow diagram of itself.

The last system illustrated is a system which provides passive control of three different aerodynamic control surfaces for a Benchmark Active Controls Testing (BACT) model in the

Transonic Dynamic Tunnel (TDT). This Passive Digital Controller System (PDCS), developed for the BACT, was used in the tunnel in November 1993 and will be used again in November 1994. It interfaces with a real-time DMA controller to command control surface positions and excitations, but does not actively employ sensor signals from the wing from which to compute control surface commands in order to suppress aeroelastic phenomena such as flutter. It does provide the following functions:

- Static command of control surfaces positions
- Excitation of control surfaces, singly or in combination
- Monitoring of control surface positions and error signals, actuator hydraulic pressures, and hinge moments
- 'Trip' System control for wind-tunnel safety

DIGITAL CONTROL OF WIND-TUNNEL MODELS USING LabVIEW

by

Sherwood T. Hoadley

**LaRC Computer Systems Technical Committee Workshop
on**

**The Role of Computers in LaRC R&D
June 15-16, 1994**

OBJECTIVE

DEMONSTRATE THE USE OF LABVIEW®

FOR

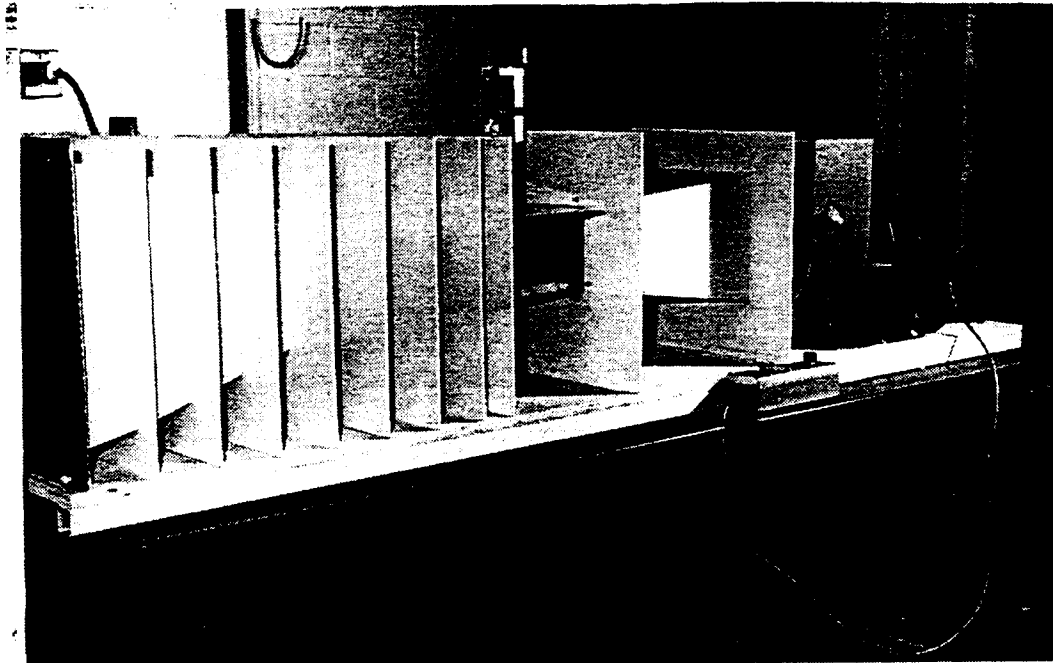
INTERACTIVE ANIMATED

DISPLAY OF ACQUIRED EXPERIMENTAL DATA

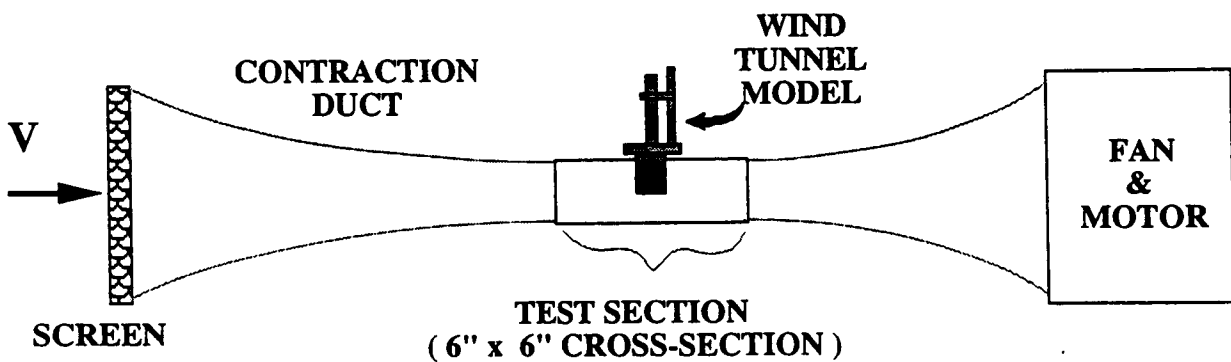
AND

REAL-TIME CONTROL OF WIND-TUNNEL MODELS

TABLE TOP WIND TUNNEL FACILITY FLUTTER RESEARCH AND EXPERIMENT DEVICE



WIND TUNNEL FACILITY FLUTTER RESEARCH AND EXPERIMENT DEVICE



CEILING MOUNTING FOR MODELS

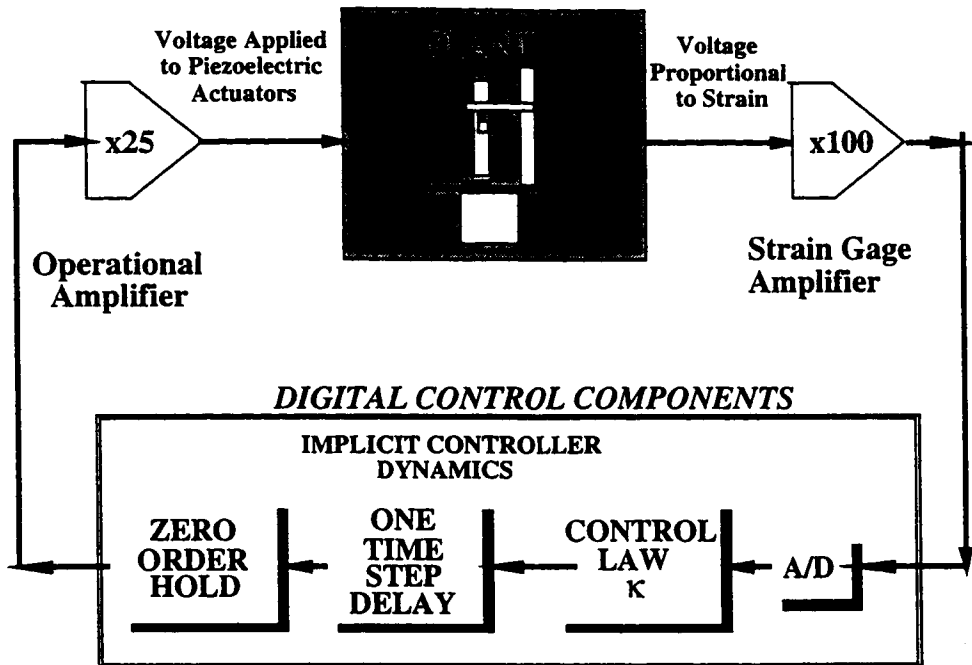
LIMITS:

VELOCITY
MACH

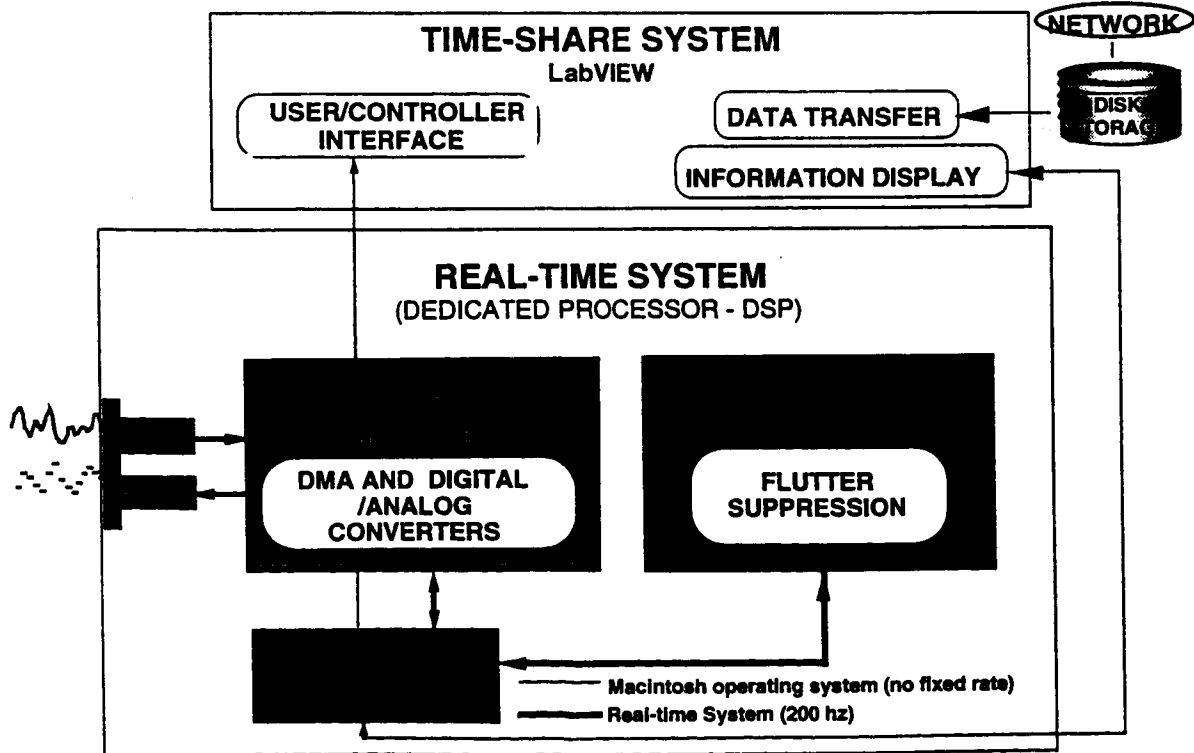
1500 inches /second

.112

BLOCK DIAGRAM OF CLOSED LOOP SYSTEM

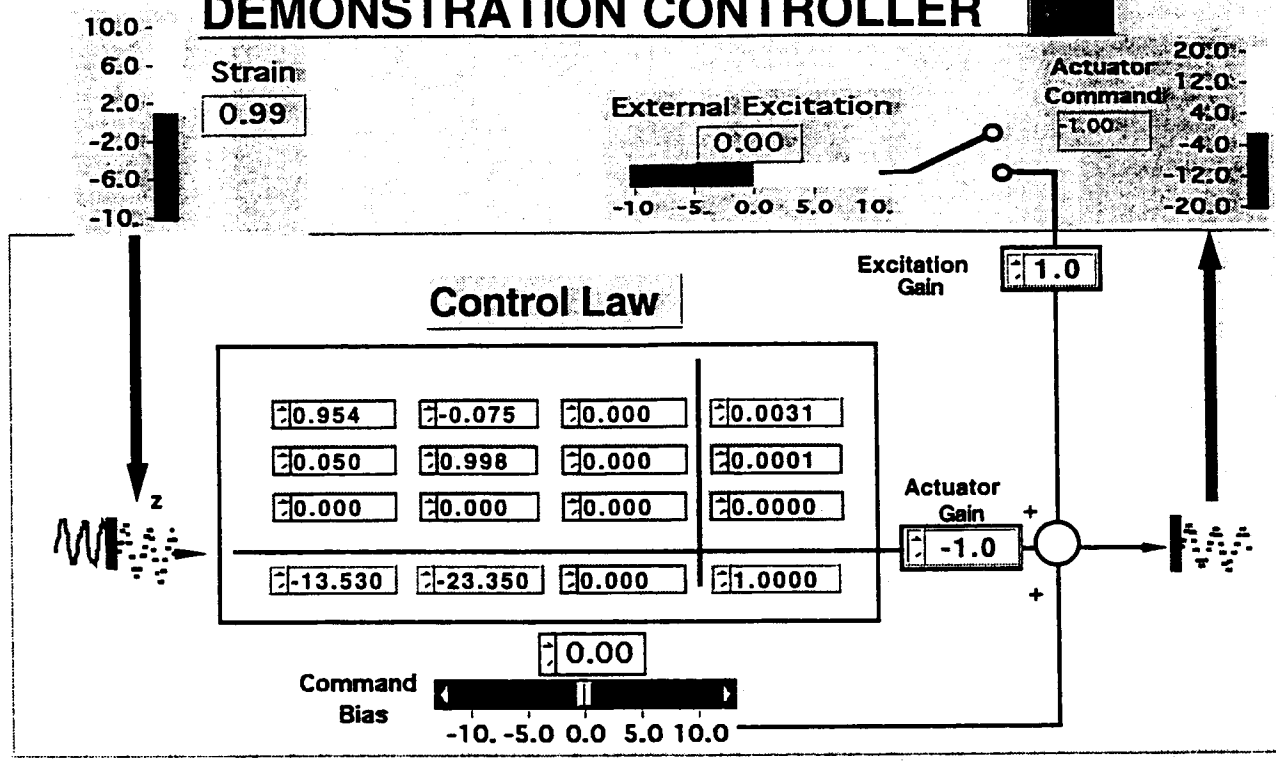


DIGITAL CONTROL SOFTWARE SCHEMATIC



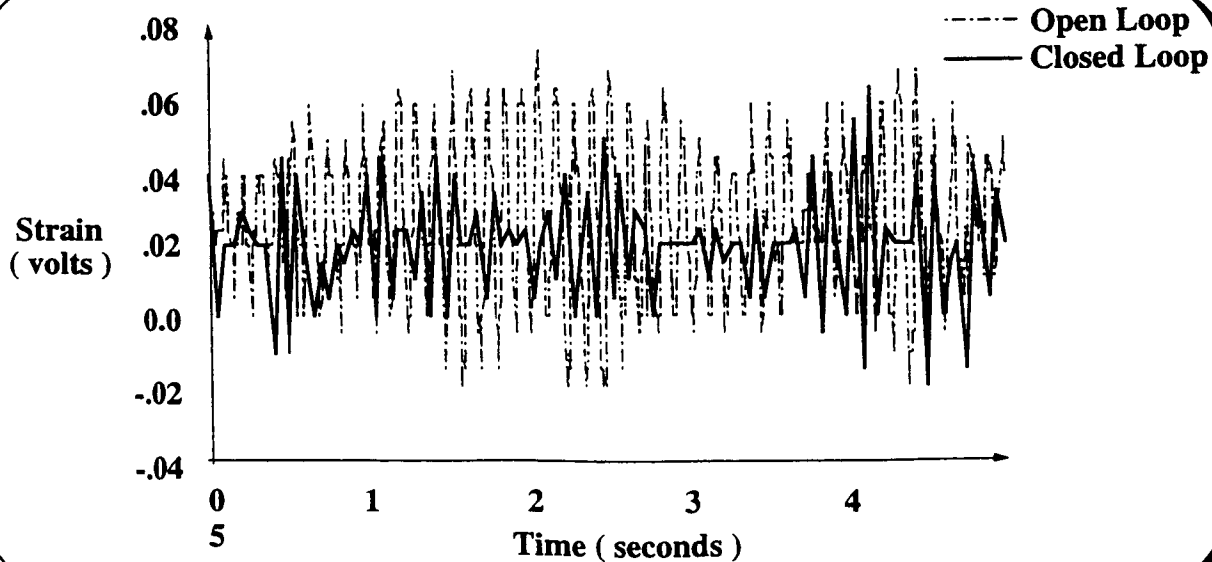
LabVIEW FLUTTER SUPPRESSION CONTROLLER VIRTUAL INSTRUMENT FRONT PANEL

DEMONSTRATION CONTROLLER



OPEN AND CLOSED LOOP STRAIN RESPONSE TO WIND TUNNEL TURBULENCE

Wind Tunnel Velocity 575 inches / second



LabVIEW DATA LOGGER VIRTUAL INSTRUMENT FRONT PANEL

Initialize Serial Port & HYDRA



Port

MODEM

Pathname

BMGQ HD:Desktop Folder:

Filename

Datalog.06/16/

Read Data



Save Data Stream



Open New Data File

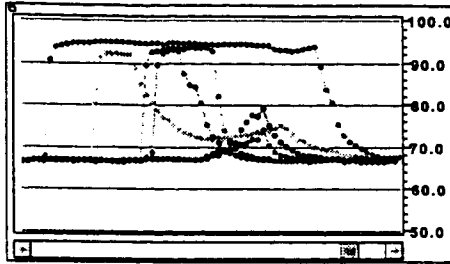


Number of Thermocouples

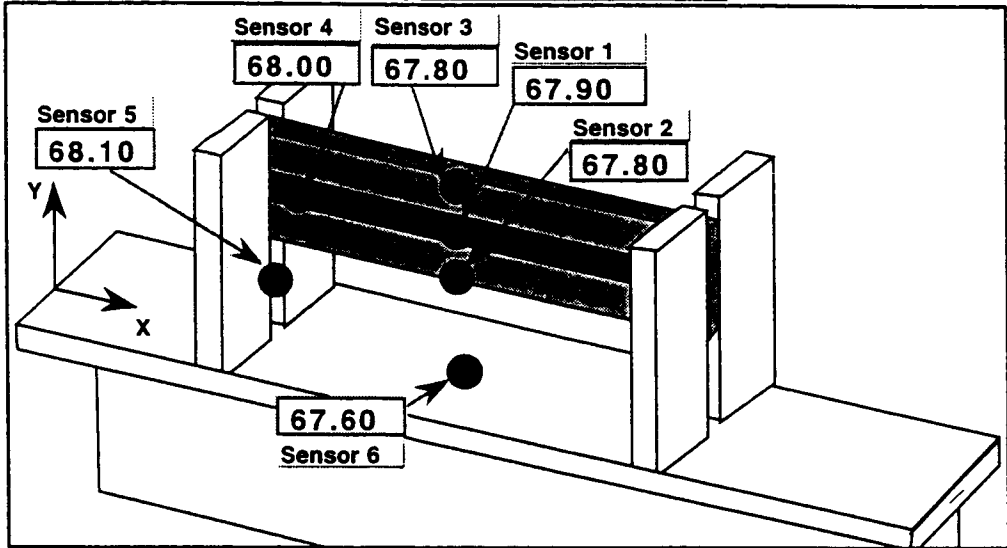
6

Delay Between Samples, sec.

2

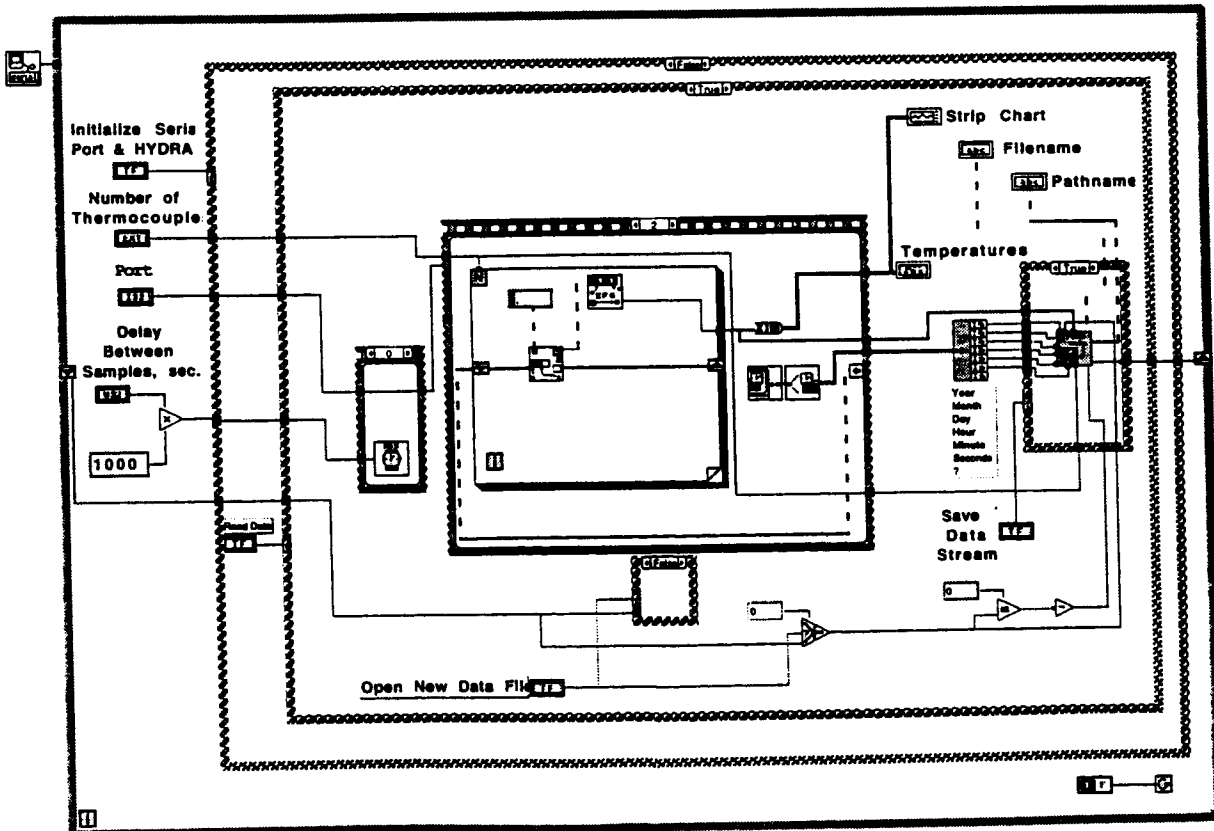


- Temp 1
- Temp 2
- Temp 3
- Temp 4
- Temp 5
- Temp 6

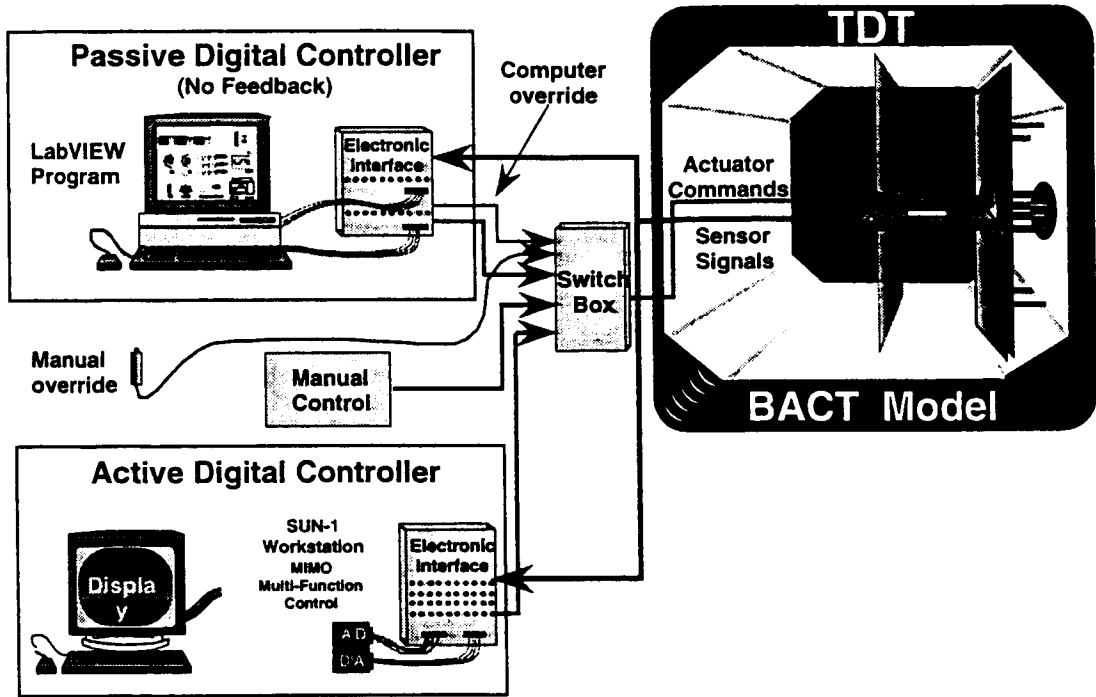


Hydra Data Logger DAQ.950
Wednesday, June 15, 1994 5:05 PM

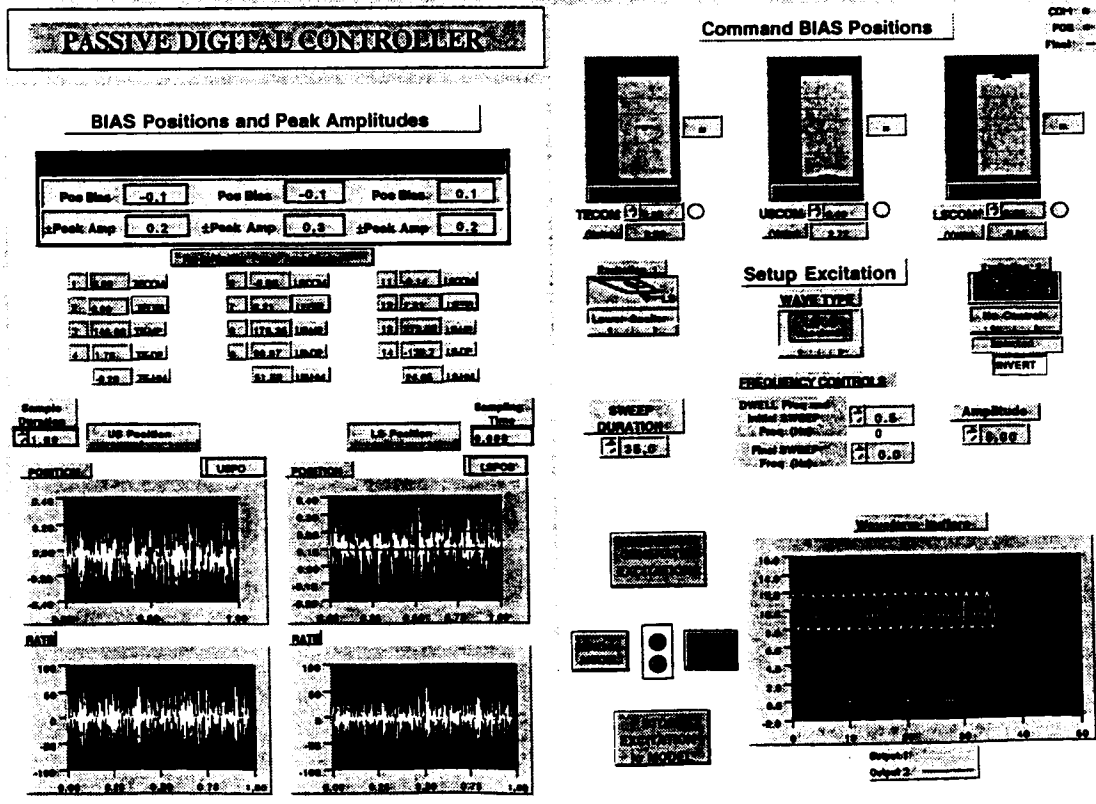
Block Diagram



BACT Digital Controller System



LabVIEW PASSIVE DIGITAL CONTROLLER & DATA ACQUISITION VIRTUAL INSTRUMENT FRONT PANEL



356170

110058

N95-16473

**Electronic Engineering Notebook: A Software Environment
For Research Execution, Documentation, and Dissemination**

P.

by Dan Moerder

The Electronic Engineering Notebook (EEN) is a byproduct of several years of collaborative work between LaRC and Martin Marietta Astronautics Group. The EEN consists of a free-form research notebook, implemented in a commercial package for distributed hypermedia, which includes utilities for graphics capture, formatting and display of LaTeX constructs, and interfaces to the host operating system. The latter capability consists of an informal Computer-Aided Software Engineering (CASE) tool, and a means to associate executable scripts with source objects. The EEN runs on Sun and HP workstations.

The EEN, in day-to-day use, can be used in much the same manner as the sort of research notes most of us keep during development of our projects. Graphs can be pasted in, equations can be entered via LaTeX, and so on. In addition, the fact that the notebook is hypermedia permits easy management of "context": e.g. derivations and data can contain easily formed links to other supporting derivations and data. The CASE tool also permits development and maintenance of source code directly in the notebook, with access to its derivations and data.

The EEN is currently in day-to-day use in the Guidance Group of the Guidance and Control Branch, and at Martin Marietta Astronautics Group.

Electronic Engineering Notebook

***LaRC CSTC Workshop
16 June 1994***

***Dan Moerder
moerder@moerder.larc***

• @Parent

Purpose of Briefing

Describe capabilities and demonstrated applications of the Notebook

Illustrate advantages and disadvantages of Notebook-based documentation over traditional approaches to research knowledge capture

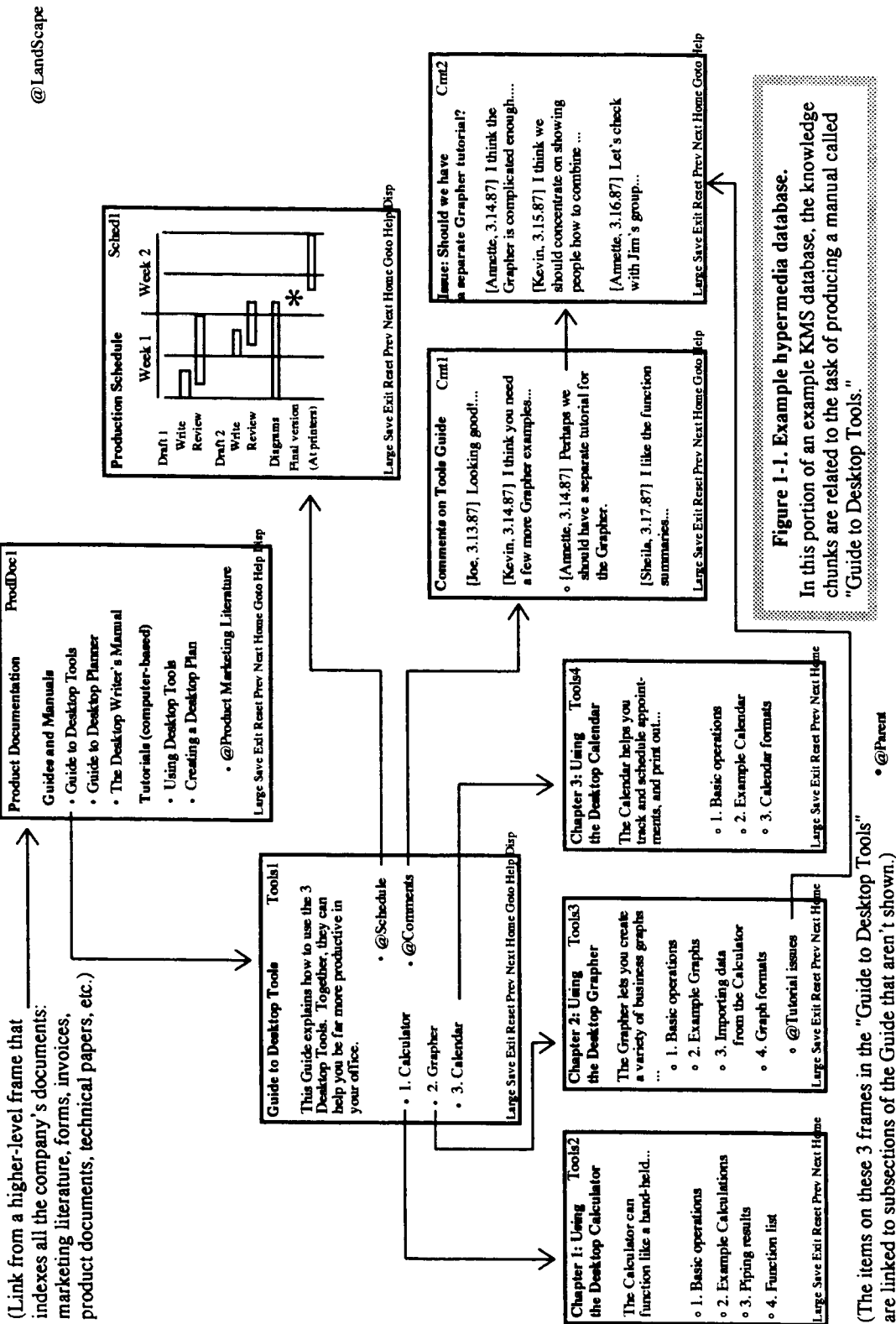
Discuss availability and status of the Notebook

What is the Notebook?

Distributed hypermedia toolset for managing results of a research team's work...

- Structured for "unruly" information***
- On the fly hypertext links between small "chunks" of information***
- Equation editing and display via LaTeX and symbol palette***
- Capture and pasting of X images, e.g. Matlab plots***
- "Informal," "researchy" CASE tool***

Hypermedia Document Example



@LandScape

Notebook Implementation

- Based on COTS software for SUN and HP workstations**
- Distributed databases are shared, enhancing team access to knowledge and codes.**
- Knowledge, includes working notes, graphics, analysis codes, and machinery for running them all form a "document" with executable components....**
- Components have been assembled, primarily, under Martin Marietta CRAD and IRAD support, with Langley involvement.**
- Notebook is a component of a Martin reusable engineering system - the Process Management Environment.**

Research Tasks in the EEN

- Recording Derivations**
- Doodling**
- Writing Code**
- Setting Up Problems for Solution**
- Managing Data**
- Doing Experiments**
- Writing A Paper**

Case History

Task: Development of a trajectory optimization code for generating reference trajectories

- Define simple class of optimal control problems to be solved***
- Establish notation for representation in parameter optimization software***
- Realize representation in FORTRAN***
- Develop test cases***
- Extend to more complicated optimal control structures***
- Write a Paper***

Statement of single-phase optimal control problem

$$\left. \begin{aligned} x^0 &\in \mathcal{R}^n \\ x^1 &\in \mathcal{R}^n \\ u(t) &\in \mathcal{R}^m \quad t \in [0, 1] \\ p &\in \mathcal{R}^r \end{aligned} \right\} \quad (1)$$

$$J = \phi(x^0, x^1, p) \quad (2)$$

$$x^0 = x(0) \quad x^1 = x(1) \quad (3)$$

$$\dot{x} = f(x, u, p) \quad (4)$$

$$\psi(x^0, x^1, p) = 0 \quad (5)$$

$$(g_{min})_i \leq g_i(x, u, p) \leq (g_{max})_i \quad 0 \leq t \leq 1 \quad (6)$$

$$(h_{min})_j \leq h_j(x, u, p) \leq (h_{max})_j \quad 0 \leq t \leq 1 \quad (7)$$

$$\dot{x} = \tilde{f}(x, u, \tau) = \tau f(x, u) \quad (8)$$

@LaTeX
 @UTLS DIR
 @MATLAB
 @XWD
 @IMAGE MGR
 @FONT

This frame states the problem that we're interested in solving. Determine

$$(1) \quad \begin{aligned} &\text{Vary } p \text{ (range) (expression)} \\ &\text{Vary } x^0 \text{ (range) (expression)} \\ &\text{Vary } x^1 \text{ (range) (expression)} \\ &\text{Vary } u \text{ (range) (expression)} \\ &\text{Vary } t \text{ (range) (expression)} \\ &\text{Vary } \tau \text{ (range) (expression)} \end{aligned}$$

to minimize

$$(2) \quad \begin{aligned} &\text{Vary } J \text{ (range) (expression)} \\ &\text{Vary } \phi \text{ (range) (expression)} \end{aligned}$$

where

$$(3) \quad \begin{aligned} &\text{Vary } x^0 \text{ (range) (expression)} \\ &\text{Vary } x^1 \text{ (range) (expression)} \end{aligned}$$

subject to

$$(4) \quad \begin{aligned} &\text{Vary } f \text{ (range) (expression)} \\ &\text{Vary } \dot{x} \text{ (range) (expression)} \end{aligned}$$

$$(5) \quad \begin{aligned} &\text{Vary } \psi \text{ (range) (expression)} \\ &\text{Vary } \psi \text{ (range) (expression)} \end{aligned}$$

$$(6) \quad \begin{aligned} &\text{Vary } g \text{ (range) (expression)} \\ &\text{Vary } h \text{ (range) (expression)} \end{aligned}$$

and the state/control inequality constraints

$$(7) \quad \begin{aligned} &\text{Vary } g \text{ (range) (expression)} \\ &\text{Vary } h \text{ (range) (expression)} \end{aligned}$$

$$(8) \quad \begin{aligned} &\text{Vary } \tau \text{ (range) (expression)} \\ &\text{Vary } \tau \text{ (range) (expression)} \end{aligned}$$

for $i=1, \dots, n_g$ and state inequality constraints

$$(7) \quad \begin{aligned} &\text{Vary } g \text{ (range) (expression)} \\ &\text{Vary } h \text{ (range) (expression)} \end{aligned}$$

for inequality constraints $j=1, \dots, n_h$. This splitting up of the constraints is due to the fact that we'll be using a state discretization in which state derivatives are calculated at the interior of discretization intervals. The controls affect state derivatives, while the state-only trajectory constraints are imposed at the discretization points.

Note that the problem is posed in Meyer form. The obvious measure for treating integral cost functions is to establish the cost function as a state. Note also that the problem is posed with unity duration. This can be generalized to free time by treating the trajectory duration as an element of the p vector, and scaling the plant ODES, e.g.

Note that (6,7) are expressed as they are because of the representation of constraints in NPSOL.

Similarly, the state inequality constraints are arranged as

$$(h_{min})_j \leq H_j \leq (h_{max})_j \quad j = 1, \dots, n_h \quad (7)$$

Note that we are not posing the problem in such a way that the upper and lower bounds for G_i and H_i are functions of the x 's or p . We are not totally comfortable with the notion of using the upper and lower bounds aggressively, since they are defined outside the logic for calculating the G 's and H 's.

The boundary conditions are restated as

$$\psi(x_1, x_{N+1}, p) = 0 \quad (8)$$

and concatenated with the system dynamics (4) to form the equality constraints

$$Z = \begin{bmatrix} z \\ \psi \end{bmatrix} = 0 \quad (9)$$

The inequality constraints (6,7) are concatenated to form

$$Y = \begin{bmatrix} (g_{min})_1 \leq G_1 \leq (g_{max})_1 \\ \vdots \\ (g_{min})_{n_g} \leq G_{n_g} \leq (g_{max})_{n_g} \\ (h_{min})_1 \leq H_1 \leq (h_{max})_1 \\ \vdots \\ (h_{min})_{n_h} \leq H_{n_h} \leq (h_{max})_{n_h} \end{bmatrix} \quad (10)$$

Note that (10) is $N \cdot n_g + (N+1) \cdot n_h$ constraints.

As a final note, the cost is expressed as

$$J = \phi(x_1, x_{N+1}, p) \quad (11)$$

The free variables in this code are arranged as

$$X^T = [x_1^T \quad u_1^T \quad x_2^T \quad u_2^T \quad \dots \quad x_N^T \quad u_N^T \quad x_{N+1}^T \quad p^T] \quad (12)$$

@LaTeX
 @UTLS DIR
 @MATLAB
 @XWD
 @IMAGE MGR
 @FONT

• Get from X Windows clipboard • Attach to X Windows clipboard
 • M-File Tool • FORTRAN Tool • Get Back to Title
 • @Default
 @MORE

This frame summarizes the details needed to run the single-phase direct NLP shooting code. The code uses NPSOL to minimize a cost function subject to plant dynamics, boundary conditions, and miscellaneous user-specified inequality constraints. Note that, for this version of the code, no interior boundary conditions, e.g. staging, are permitted. In addition, the problem is assumed to be cast in Meyer form, with unity duration. The problem statement is

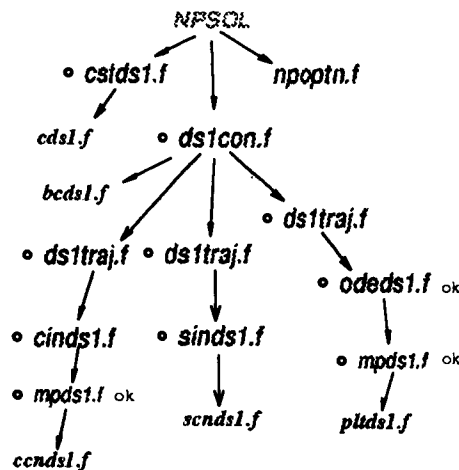
• **Problem Statement Here**

and its representation in the code is laid out here:

• **Derivation Here...**

In order to run the code, the user supplies a main routine, and five subroutines: cost, boundary conditions, RHS of the plant ODE's, state inequality constraints, and control inequality constraints. Templates for these routines are given below:

- **Template for the main routine, mads1.f**
- **Template for cds1.f (Cost Function)**
- **Template for bcds1.f (Boundary Conditions)**
- **Template for pltds1.f (RHS of plant ODEs)**
- **Template for scnds1.f (State Inequality Constraints)**
- **Template for ccnds1.f (State/Control Inequality Constraints)**



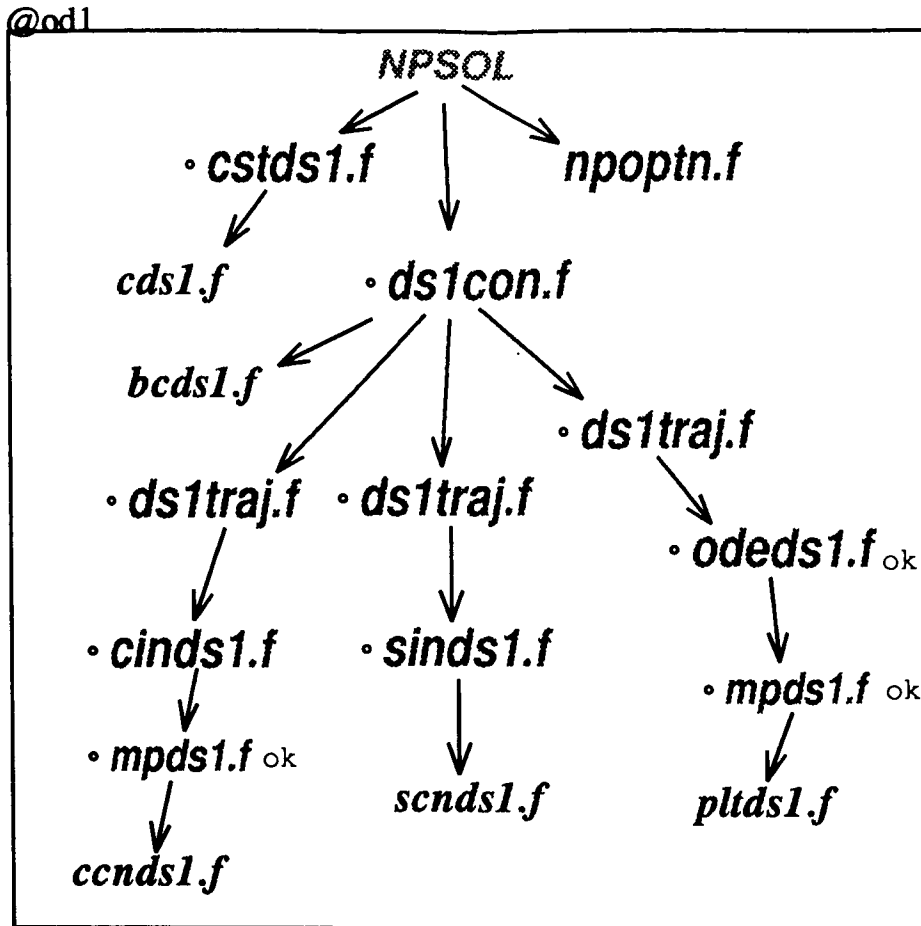
• NPSO

@LaTeX
 @UTILS DIR
 @MATLAB
 @XWD
 @IMAGE MGR
 @FONT

• M-File Tool • FORTRAN Tool • Get Back to Title

• @Parent
@MORE

Set up the code structure



- @LaTeX
- @UTILS DIR
- @MATLAB
- @XWD
- @IMAGE MGR
- @FONT

• M-File Tool • FORTRAN Tool • Get Back to Title

• @Parent
@MORE

PARAMETERS

TOGGLE

FREEZE

EXPORT

EXECUTE

• @EXECUTE SCRIPT

@

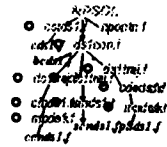
This is a template for mads.f, which calls NPSOL for doing single-phase discretized direct NLP. Here's a map of the code:

C

```

program mads1
implicit double precision(a-h,o-z)

```



• C***Declarations and User-Defined Parameters

```

@ NPLNTP NROWA maxpval MAXWKP
  NUP    NROWJ maxwk   ndint
        NROWR nu      nis
        nplnt niu
        nbc
        nparms

```

• C***User Defines Logic for getting plant parameters and initial state and control guess

```

@ NCNLN NROWA maxpval ndint @
  NCNLIN NROWJ maxwk   nis  parms
  NCON   NROW  nu      niu   x
  N      MAXWKP nplnt
  LWORK          nbc
  LIWORK         nparms

```

@ parms=plant parameters
x=initial guess for state, control history and parameters.

- cds1.f**
- bcds1.f**
- pltds1.f**
- ccnds1.f**
- scnds1.f**

• C***Execute NPSOL

• **makefile**

```

@
x=solution?
c=final value of constraint vector.

```

• C***User Defines logic to save results

C

```

stop
end

```

• Clone this frameset!

• @Parent

C***Set NPSOL execution parameters

PARAMETERS

TOGGLE

FREEZE

EXPORT

EXECUTE

• @EXECUTE
SCRIPT

@Calculate derivatives numerically

```
c
  call npopn('derivative level          0')
  call npopn('difference interval    .0000001')

c***Diagnostic
  call npopn('print level            21')
```

@Hardwired interval for numerical differentiation. If this isn't used, NPSOL will waste time figuring this out on a case-by-case basis.

HELP

• @Parent

Last file export on: 28 January 94 at 10:49:57, current version

- Info • **Top frame of tree to write:** todi0001a2
- Info • **File:** /moerder/usr1/moerder/TODI-II/ds1/mads1.f
- Info • **Add blank line between items:** yes
- Info • **Follow tree items linking to other framesets:** no
- Info • **Follow annotation items linking within the frameset:** no
- Info • **Time version number:**
- Info • **Preserve relative indentation of items:** no
- Info • **Template frame:**
- Info • **Remove 1st character of each line during export:** no

- Info • **Program to execute script:** shell script

- Info • **Toggle text 1, family:** Times
- Info • **Toggle text 1, size:** 16
- Info • **Toggle text 2, family:** Courier
- Info • **Toggle text 2, size:** 14

• ***This script initially cloned from 'shoot0019a' 27 December 93 10:33:32***

• @Parent

@Top of startup script

cd /moerder/usr1/moerder/TODI-II/fighter

```
f77 -O2 -o mads1 mads1.f cds1.o bcds1.o pltds1.o ccnds1.o scnds1.o \  
getprm.o ficof.o \  
../ds1/ds1con.o \  
../ds1/cstds1.o \  
../ds1/ds1traj.o \  
../ds1/odeds1.o \  
../ds1/cinds1.o \  
../ds1/sinds1.o \  
../ds1/mpds1.o \  
-lnpsol -llinpackd
```

•@Parent

This note lays out the aircraft model paraphrased from Hans Seywald's thesis . The equations of motion are

$$\dot{v} = (T - D) \frac{\cos \gamma}{m} - g \sin \gamma \tag{1}$$

$$\dot{h} = v \sin \gamma \tag{2}$$

$$\dot{\gamma} = \frac{g}{v} (n - \cos \gamma) \tag{3}$$

$$\dot{x} = v \cos \gamma \tag{4}$$

$$\rho(h) = \frac{1.225}{g} e^{-\gamma h} \tag{5}$$

$$y = (ay)_1 + (ay)_2 h + (ay)_3 (e^{-\gamma h}) \tag{6}$$

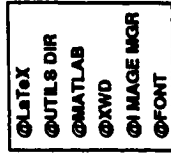
$$z(h) = \sum_{j=1}^4 (az)_j h^j \tag{7}$$

$$a(h) = 20.0468 \sqrt{\theta} \tag{8}$$

$$\theta(h) = \sum_{j=1}^4 (\alpha\theta)_j h^{j-1} \tag{9}$$

$$q = \rho v^2 / 2 \tag{10}$$

• @MORE



• Get from X Windows clipboard • Attach to X Windows clipboard
 • M-File Tool • FORTRAN Tool • Get Back to Title

• @Handscope @FORMAT @STYLES • @Type:FormattedDoc

• Miscellaneous Constants

• ay

@These are constants for the Hans F15 model

• az

• aθ

• acd0 • bcd0

• *Drag Model*

• ak • bk

• aε

• @Parent

PARAMETERS

TOGGLE

FREEZE

EXPORT

EXECUTE

• @EXECUTE
SCRIPT

~~@+7.29821847445e-1
-3.25219000620
+5.72789877344
-4.57116286752
+1.37368651246~~

+1.37368651246
-4.57116286752
+5.72789877344
-3.25219000620
+7.29821847445e-1

HELP

• @Parent

Flight Envelope Calculations

$$\begin{aligned}
 v^+(h) &= \arg \max_h v \\
 v^-(h) &= \arg \min_h v \\
 T(h, v) - D(h, v) &= 0 \\
 \gamma &= 0 \\
 n &= 1
 \end{aligned}
 \tag{1} \tag{2} \tag{3}$$

This note lays out the calculations for generating a flight envelope for the fighter aircraft model. In setting this up, I'm assuming that the envelope is convex. Pursuant to this assumption, we get roughly half of the envelope by solving

```

\Large\begin{equation}
\text{\texttt{\scriptsize}}
\end{equation}

```

subject to

```

\Large\begin{equation}
\text{\texttt{\scriptsize}}
\end{equation}

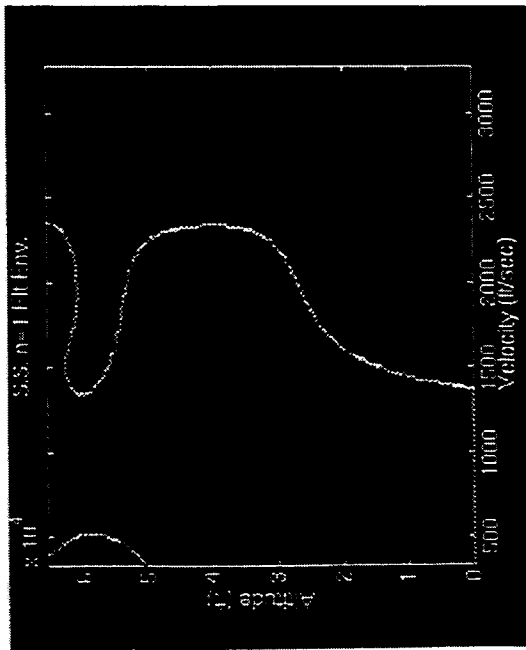
```

for a given γ and load factor, in this case.

Code Using NPSOL and the other machinery for this example

This doesn't seem to work out. Here's a check of the thrust and drag models.

Code to display this stuff • Code to Check T-D over flight regime



```

@LtoX
@UTILS DIR
@MATLAB
@XWD
@IMAGE MGR
@FONT

```

• Get from X Windows clipboard • Attach to X Windows clipboard
 • M-File Tool • FORTRAN Tool • Get Back to Title • @Parent
@MORE

Naturally, The NPSOL business hasn't converged except in a very narrow range of conditions. I'm now checking against Hans' model to determine where our numbers diverge.

Code for Checking against Hans

The flight envelope to the left corresponds to the case where all conflicts with Hans' model have been resolved, with the exception of the Drag expression.

• **Diary for this problem (and) How to find Dan's F-15 solution !!!**

Do each of these things just in case GEORGE has dropped a disk !!

- (1) **Matlab script to find the "slopy" F-15 data and save.**
- (2) **Do some FORTRAN and create the necessary jacobians.**
- (3) **Costate approx function for slopy F-15 -- super easy...**
- (4) **Use approx costates as init guess in shooting...**

• Min-time-to-climb with one control segment

@LaTeX
@UTILS DIR
@MATLAB
@XWD
@IMAGE MOR
@FONT

• Get from X Windows clipboard • Attach to X Windows clipboard
 • @M-File Tool • @FORTRAN Tool • @Get Back to Title • @Parent
@MORE

@landscape @FORMAT @STYLES • @Type:FormattedDoc

costappii77

Dans' solution to the F-15 mimimetoclimb problem with one control seg can be found in /moerder/usr1/moerder/TODI-II/slop_fighter/e2000_38000_21.dat !!!

This file has 21 individual problems (each with a different terminal energy). Each run has 20 nodes, 6 boundary conditions, 5 states, 1 (phoney) control, and 3 free params (that define the actual control found in x(4)) thus each solution has 128 rpsol params.

20 nodes * 5 states + 19 (ctris appear at midpoints) + 6 bc's + 3 free params = 128

Anyway the data we want is found in (2561:2688); that is the last 128 lines in the above file !!!

It took forever to figure this out !!!!!

Next we must remember that Dan's data does not have the control at the midpoints but at the node points. I will use midpoint_filter.m to correct this.

Grrrrr.....

Last but not least Dan's data does not have the time state as needed by my code. I will correct for this in the usual way.

Grrrrr..... Irritation -- like a rash that won't go away...

All of the above is taken care of in the (1) matlab script on the previous frame... O.K. I did this... and the result is stored in xu_slop.dat in directory -- TODI-II/fighter

Next the trajectory jacobians are calculated in fortran !!! O.K. I did this... and the resulting files are stored in xu_slop_fjac.dat and xu_slop_f.dat in TODI-II/fighter

@LaTeX
@UTLS DIR
@MATLAB
@XWD
@MAGE MGR
@FONT

• Get from X Windows clipboard @Present
• Attach to X Windows clipboard
• @M-File Tool • @FORTRAN Tool • @Cat Back to Title @MORE

• %Header

• %Introduction

• %Single-Impulse Problem Statement @FIG 4

• %Results @FIG 9

• %Two-Impulse Problem Statement

• %Figures and Tables

• % Bibliography

\end{document}

• @Parent

%Single-Impulse Problem Statement

PARAMETERS

TOGGLE

FREEZE

EXPORT

EXECUTE

• @EXECUTE
SCRIPT

\section{Vehicle Model and Mission Description}

• % Vehicle Model

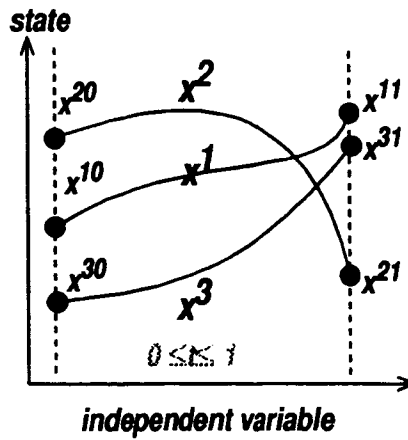
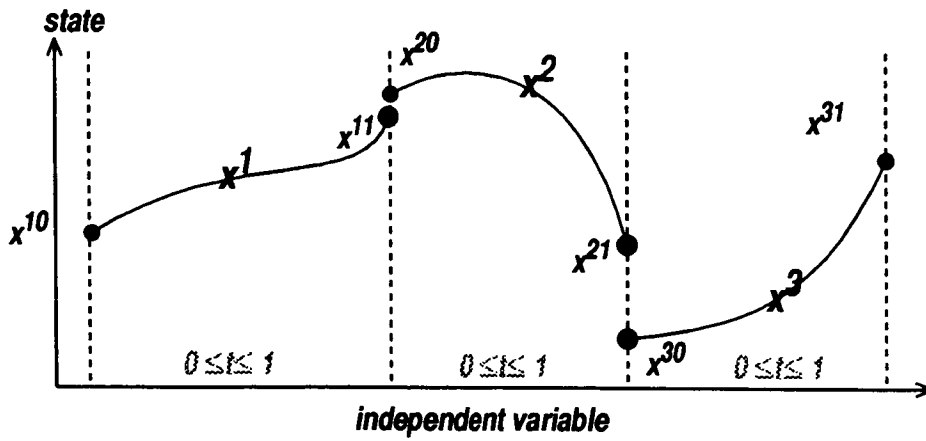
• % Mission Parameters

• % Constraints

• % Optimization Formulation

HELP

• @Parent



@LaTeX
 @UTILS DIR
 @MATLAB
 @XWD
 @MAGE MGR
 @FONT

• M-File Tool

• FORTRAN Tool

• Get Back to Title

• @Parent
 @MORE

• Link to

Current Status

- *Notebook system has been (stoically) tested and commented on by GCB's Guidance Group for almost a year.*
 - *Testing and comments have resulted in large changes in interface; notebook is going into its 4th major revision.*
- *Martin Process Management Environment (and our Notebook) suffered from very troublesome learning curve - both being "dumbed down" for usability under IRAD funding.*
 - *Martin has developed a no-cost licensing agreement for distribution of the new system to non-NASA users.*
- *The host software vendor (Knowledge Systems), in response to loud and persistent suggestions from Martin and Langley will distribute read-only licenses free of charge, and full licenses free to academic institutions.*
 - *This is favorable for technology interchange...*

Summary

- *The Notebook does capture and organize the work of research teams.*
 - *It is in daily use by 5 researchers at LaRC*
 - *And by roughly 40 engineers at Martin (in its PME form).*
 - *And, being baselined in a proposed university multidisciplinary design curriculum.*
- *The Notebook is very useful in its current form, and has fewer irritating features with each revision.*
- *Difficulties with distribution of notebook-capture knowledge are being resolved, e.g. licenses.*
- *The Notebook is currently available to Langley researchers with Sun or HP workstations. Moerder will gladly discuss this in more detail offline, set up live demos, etc.*

IDEAS² Computer Aided Engineering Software by Pat Troutman

IDEAS² is a multidisciplinary Computer Aided Engineering (CAE) software tool that was developed for systems engineering and integration analysis of spacecraft. The name IDEAS² was derived from the two software packages that were integrated to form the tool. Interactive Design and Evaluation of Advanced Spacecraft (IDEAS) was a NASA spacecraft-specific analysis software tool that was combined with a commercially available product called Integrated Design Engineering Analysis Software (I-DEAS). I-DEAS is a Structural Dynamics Research Corporation (SDRC) product that provided capabilities lacking in NASA IDEAS such as solid and finite element modeling, thermal analysis and advanced graphics.

IDEAS² utilizes a common database structure which facilitates the integrated flow of data between the various analysis modules. All analysis is based on information derived from a three dimensional solid math model that is created in the commercial solid modeling program. The combination facilitates traceability and ensures all analysis is based on the same information. Once the model has been generated and stored in the common database, a wide range of analysis can be performed. IDEAS² has several orbital dynamics modules that can simulate/analyze spacecraft characteristics such as controllability in the presence of dynamic operations (solar array articulation, robotic arms, etc.), orbit lifetime/reboost requirements and micro gravity environment. Structural analysis capabilities are also available ranging from finite element modeling to forced response analysis. The impact of the local spacecraft environment can also be evaluated by utilizing the IDEAS² thermal and plume impingement analysis capabilities.

The common database and integrated analysis environment allow IDEAS² to be used both for high level short term studies and large program systems integration. Several NASA centers utilize the software for advanced concept analysis dealing with space platforms or Lunar/Mars exploration. The Space Station Freedom program has established IDEAS² as its primary Level II integration software package. IDEAS² models are commonly used to disseminate the latest Freedom element weights and configuration updates. IDEAS² has recently been upgraded to allow the entire software package to be ported to a UNIX workstation along with a new graphical user interface. This will allow smaller organizations to utilize the IDEAS² capability without a significant investment in computer hardware.

IDEAS² was initially developed from 1985 to 1986 and has continuously been enhanced to include the most up to date analysis tools and graphics interfaces



IDEAS²

Computer Aided Engineering

Software

Pat Troutman
Spacecraft & Sensors Branch
LaRC Space Systems & Concepts Division



Outline

- IDEAS² Background**
- Current Capabilities**
- Lessons Learned /
Future Directions**
- IDEAS² Analysis
Simulation Video**



**NASA
IDEAS**

**Interactive
Design and
Evaluation of
Advanced
Spacecraft**

+

**SDRC
I-DEAS**

**Integrated
Design
Engineering
Analysis
Software**

= IDEAS²



What is IDEAS²

- **A multidisciplinary software tool that was developed for spacecraft analysis**
- **All analysis is based on information derived from a three dimensional solid math model that is stored in a common database structure**
- **Analysis capabilities include orbital dynamics simulation, structural modeling and analysis, thermal analysis, plume impingement and orbital debris impact analysis**



Satellite Systems Analysis Disciplines

System Design
 Databases
 Expert Systems
 Design Modules
 Costing

Function Driven

System Simulation
 System Resource
 Interaction

Physical Design
 IDEAS Modeling
 Wavefront

Geometry Driven

Physical Analysis
 IDEAS Squared
 EWB
 NASTRAN
 Sensor Analysis

Mission Analysis
 Satellite Tool Kit
 DECAT
 Orbital Workbench
 Costing

Operational
 In Development

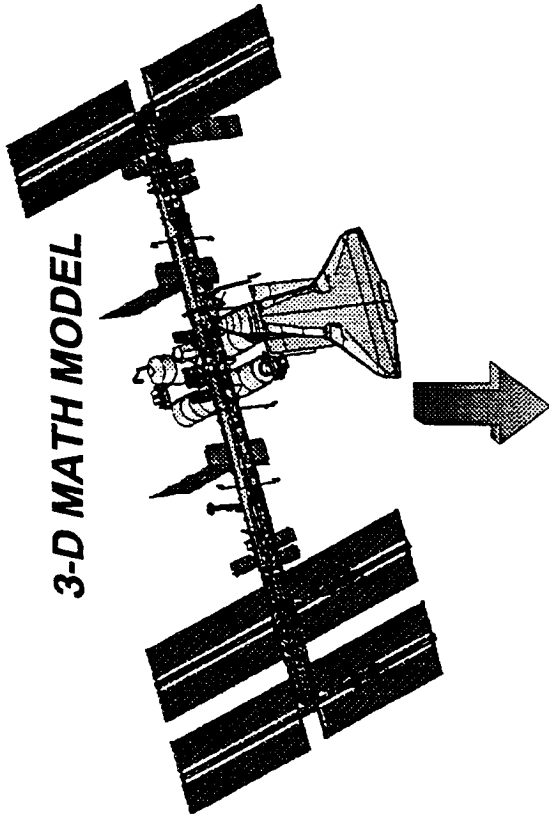


IDEAS² History

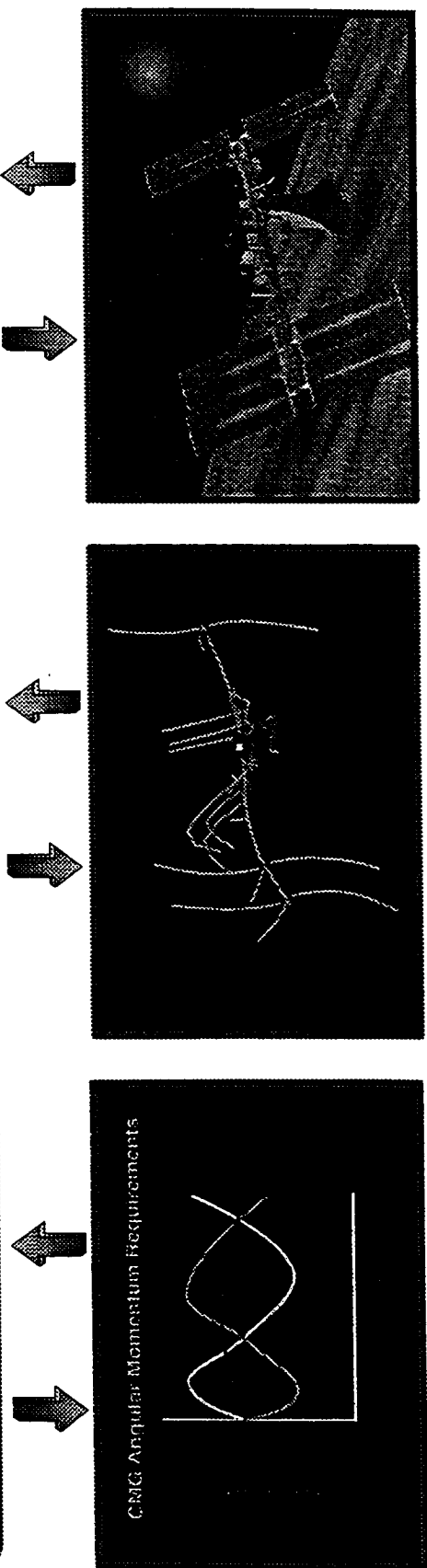
- Initially developed for the Space Station program from 1985 to 1986**
- Selected as Space Station level II integration package in 1987**
- Used by LaRC, Johnson Spaceflight Center and the University of Colorado**
- Updated in 1993 to run on a Silicon Graphics workstation with a X window graphical user interface**

IDEAS²

*Computer
Aided
Engineering*



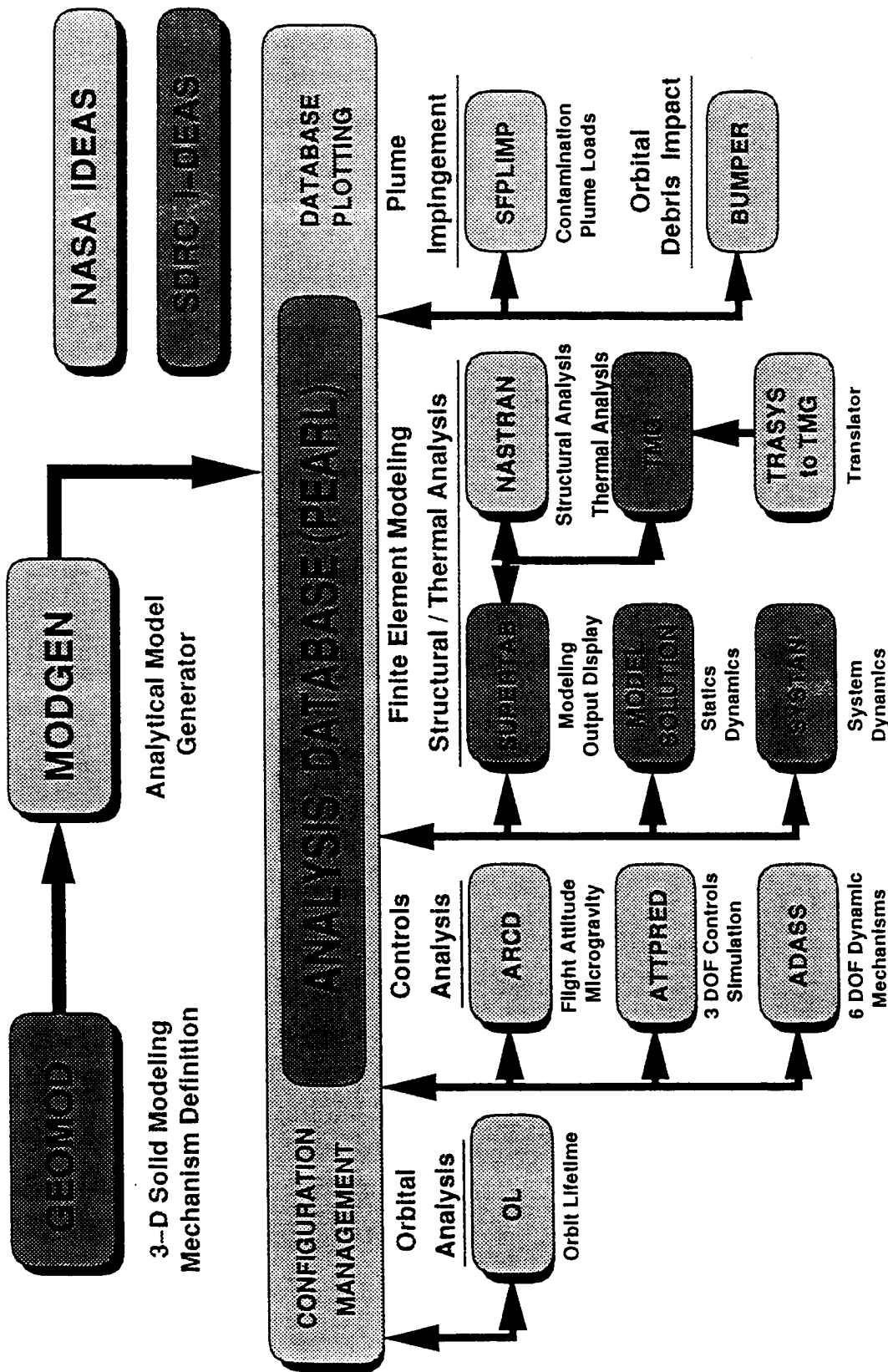
COMMON ENGINEERING DATABASE



ORBITAL DYNAMICS STRUCTURAL ANALYSIS ENVIRONMENT IMPACT



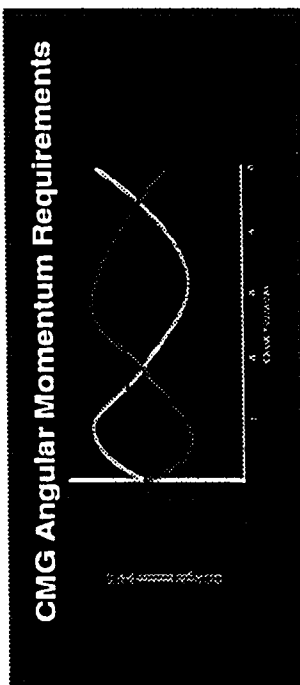
IDEAS² Capability





Spacecraft Dynamics and Control Capabilities

- 3 and 6 DOF analysis programs, passive and active control
- Attitude Control Law Simulation Capabilities
 - *CMG (attitude or momentum emphasis; momentum management)*
 - *Reaction Control System (RCS Jets)*
 - *Reaction Control Wheel (with supplemental magnetic torque rods)*
 - *Passive Magnetic Dampers*
- Microgravity Environment Determination
- Optimal Attitude Determination Capability
- Orbit Lifetime Analysis
- Reboost Guidance/Optimization Simulation Capability
- Robotic Dynamic Simulation
- Station Keeping Fuel Estimation
- ACRV Escape Trajectory clearance determination





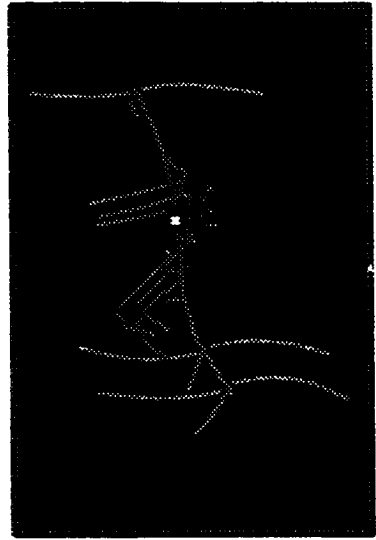
Structures/Mechanisms Analysis Capabilities

I-deas - Solid model generation (GEOMOD), finite element model generation (SUPERTAB), post processing (SUPERTAB), transient response analysis (Model Solution/SYSTAN).

ADAMS/ADASS - Mechanism/deployment analysis

NASCON - NASTRAN to SUPERTAB conversion

NASTRAN - Calculation of modes and frequencies.





Thermal / Plume Impingement / Orbital Debris Analysis Capabilities

Thermal Analysis:

I-deas - Extensive interactive geometric, finite element & finite difference modeling. Interface to "standard" thermal and structural analysis tools. **TMG** - Solar and planetary heat flux calculations.

Plume Impingement Analysis:

SFPLIMP - Designed to provide an assessment of the effects of jet firing on nearby space structures. The software allows the user to determine the instantaneous pressure loads, heating rates and contamination rates on surfaces due to jet exhaust.

Orbital Debris Analysis:

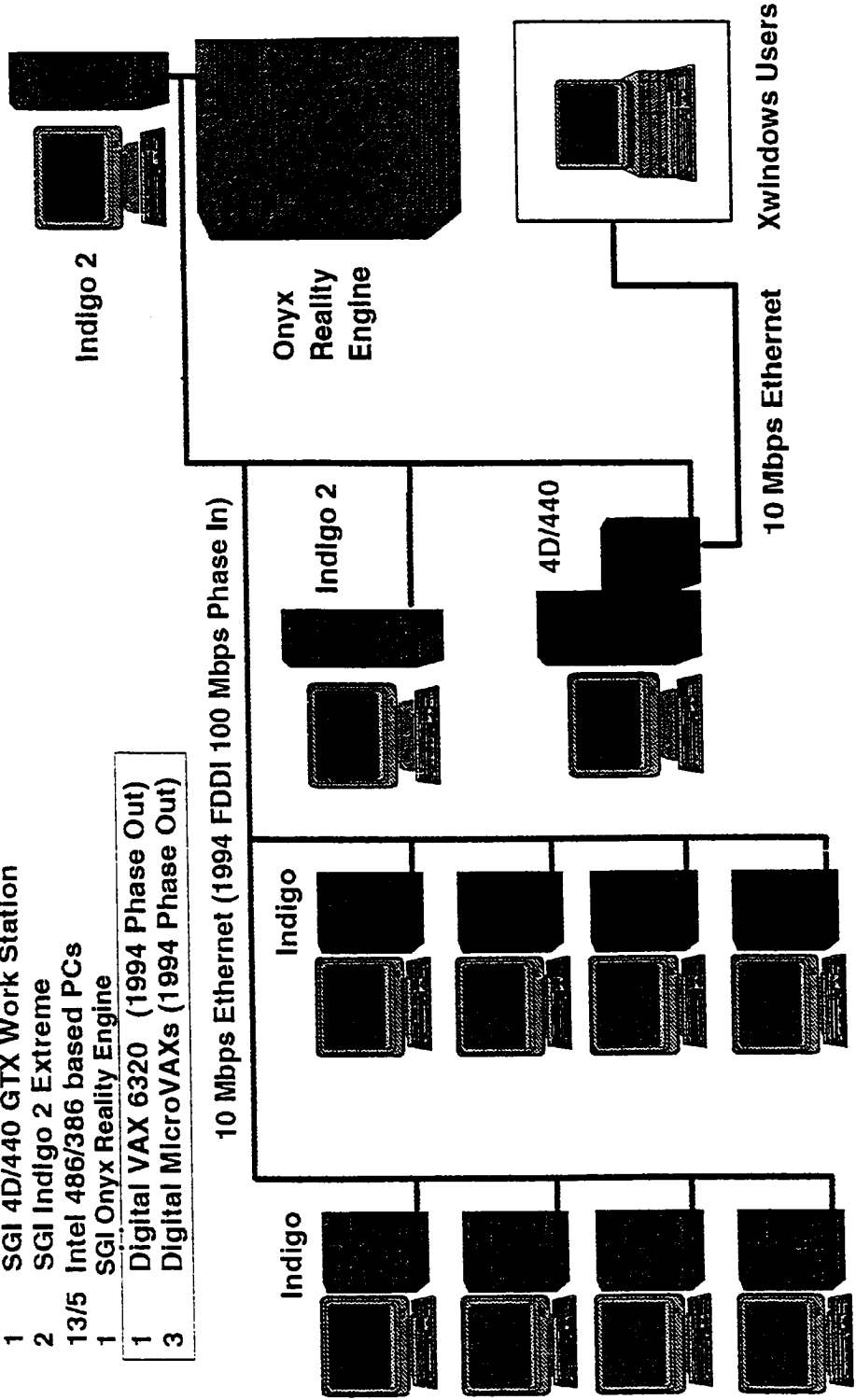
Bumper II - Predicts the probability of no penetration or no impact for spacecraft subject to man-made orbital debris or meteoroid impact. The code accounts for varying impact velocity, impact angle, wall configurations, and the effects of spacecraft geometry and orientation.



S&SB Computer System Hardware

SYSTEMS:

- 8 SGI R4000 INDIGO Work Stations
 - 1 SGI 4D/440 GTX Work Station
 - 2 SGI Indigo 2 Extreme
 - 13/5 Intel 486/386 based PCs
 - 1 SGI Onyx Reality Engine
- | | |
|---|------------------------------------|
| 1 | Digital VAX 6320 (1994 Phase Out) |
| 3 | Digital MicroVAXs (1994 Phase Out) |





Lessons Learned in Developing & Maintaining IDEAS²

Integration vs Interfacing:

IDEAS squared currently has a high degree of integration between the commercial software, the database and the NASA developed analysis codes. Changes in one area can ripple through to the others causing a maintenance backlog.

Analytical Module Development:

Engineers are hampered during analytical software development by complex database and GUI interfacing



Sample IDEAS² Analysis Video

Analysis Task: Verify that the Shuttle RMS can rotate the an early stage of the Space Station through two 90 degree rotations in one orbit's time.

Geomod – Used to build model of shuttle/station configurations

ARCD – Used to establish initial and final configuration flight attitudes

ATTPRED – Used to establish initial and final configuration stability characteristics

ADASS – Used to perform free drift/robotics simulation

Matlab as a Robust Control Design Tool

Irene M. Gregory
Dynamics and Controls Branch

This presentation is geared towards introducing Matlab as a tool used in flight control research. The example problem used to illustrate some of the capabilities of this software is a robust controller designed for a Single-Stage-To-Orbit airbreathing vehicle's ascent to orbit. The details of the problem and the control law design are available from reference 1. The global requirements on the controller are to stabilize the vehicle and follow a trajectory in the presence of atmospheric disturbances and strong dynamic coupling between airframe and propulsion. Hence, the need for a robust controller.

Matlab is an interactive program designed for numerical computation, data analysis and visualization as well as a philosophy of open architecture. Fundamentally, Matlab is built upon a foundation of sophisticated matrix software for analyzing linear systems of equations. The relevance of this is that matrices are useful because they can describe so many things in a mathematically efficient and highly flexible way. Matlab serves as a kernel from which several toolboxes are linked. Application toolboxes as the name implies are a collection of predefined functions intended to solve more application-specific problems such as a control design problem that requires system modeling, controller synthesis and analysis. One of the most important tools for modeling complex nonlinear systems and simulating them is Simulink. An example of such a system is presented here. The model consists of an integrated aerodynamics/propulsion database, various information for use with a pilot on approach and landing, and a full 6 d.o.f. rotating earth equations of motion along with an atmospheric model. Not only does Simulink provide a straight forward way to easily build this system, but it also incorporates files written in different languages, in this case FORTRAN and C, in the model without any modifications.

Given the nonlinear system we proceed with trimming the vehicle and deriving linear models. Both of these are predefined functions that can be executed in a single line. Since the system is unstable, the controller is required to both stabilize the vehicle and follow the prescribed trajectory. Typically synthesizing a controller is an iterative procedure and it becomes advantageous to automate the process. An m-file consisting of Matlab commands can be used to define scaling for optimized variables, construct the new linear system that includes these scaling, and perform controller synthesis and analysis. This system model would also include uncertainty that may arise from various physical considerations. Once written, the iterative process can be completed in a few key strokes per iteration. These m-files serve as an example of yet another language that can be utilized in Matlab.

This controller example utilizes some modern control techniques that are available from μ -tools and to some extent from Robust control toolbox. The controller synthesis problem is solved using H^∞ optimization and analysis are performed using μ , also known as structured singular value. μ is analogous to Bode plots in the classical control methodologies. The μ plot allows an immediate assessment of whether a controller has fulfilled the specified requirements. Once the desired controller has been found, a model reduction, to reduce its dynamic order, is performed using a number of techniques among them Hankel singular values and residualized truncation. A reduced order controller is then integrated into the nonlinear simulation. Time response of the system is evaluated as both a confirmation of frequency domain μ analysis and another way of evaluating results.

Matlab/Simulink combination also has the capability of automatically generating C code for any block in a diagram. This capability is very useful for transferring controller from the design environment into non-Matlab environments such as real time simulation or even flight test. These capabilities are being currently evaluated.

In summary, a number of different capabilities of Matlab were illustrated in this example. We find Matlab a powerful yet very flexible tool to use in controls research.

References:

Gregory, Irene M.; McMinn, John D.; Chowdhry, Rajiv S. and Shaughnessy, John D.: Hypersonic Vehicle Model and Control Law Development Using H^∞ and μ -Synthesis. NASA TM-4562, July 1994.

Matlab as a Robust Control Design Tool

**Irene M. Gregory
Dynamics and Controls Branch**

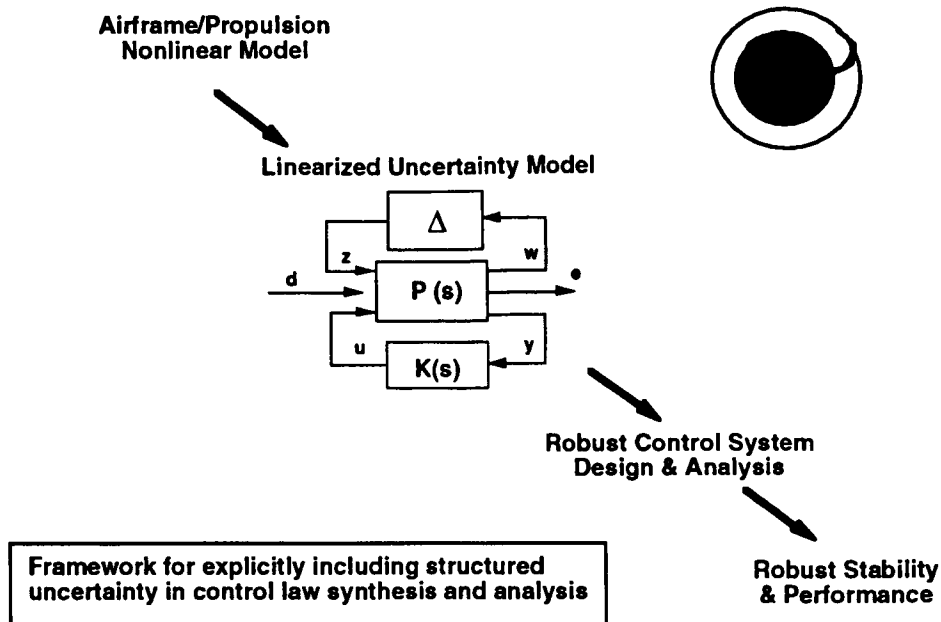
**presented at
The Role of Computers In LaRC R&D
1994 Workshop**

June 15 - June 16

Presentation Outline

- **Robust control law problem**
- **Introduction to Matlab**
- **Nonlinear system simulation**
- **Linear model derivation**
- **Sample command file**
- **Controller synthesis and analysis**
- **Concluding remarks**

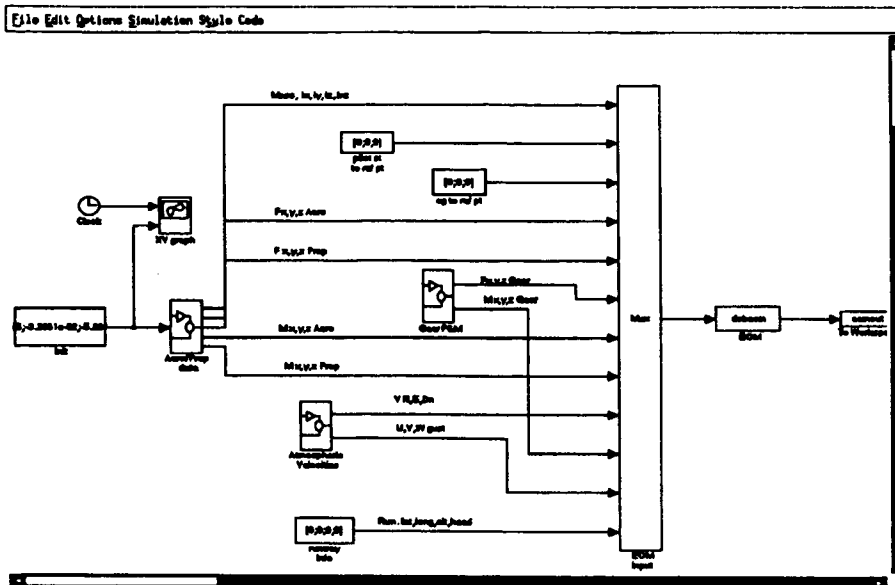
Robust Control Law Framework



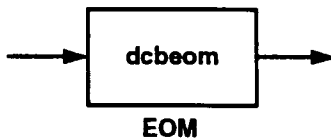
Introduction to Matlab

- **Matlab**
 - For numeric computation, visualization, and data analysis
 - The basis of Matlab is matrix manipulation and matrix solving.
 - Matlab is a kernel from which several toolboxes, a collection of predefined functions, are linked
- **Simulink**
 - For advanced nonlinear modeling and simulation
- **Application Toolboxes**
 - For customizing your Matlab environment with special tools to solve more application-specific problems.
 - » e.g. Controls, μ -Tools, Signal Processing, Neural Nets

Matlab Nonlinear Simulation



Eqn's of Motion Block

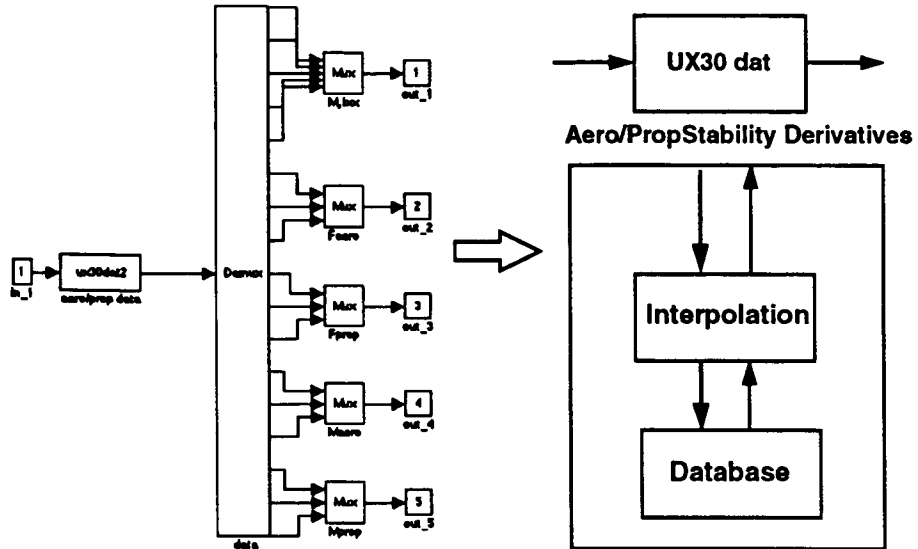


Block name: EOM	Done
Block type: (Mask)	
Subsystem: sys=func(x,u,flag,param1,...)	Revert
Subsystem function name: dcbeom	Help
Function parameters:	

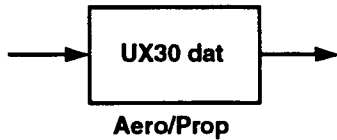
```

* mdlInitializeSizes - initialize the sizes arr
* The sizes array is used by SIMULINK to
  determ
* characteristics (number of inputs, outputs, s
  y
#define EOMDEBUG
#define NSTATES 7
#define NOUTPUTS 115
#define NINPUTS 39
static void mdlInitializeSizes(S
  SimStruct *S;
{
  ssSetNumContStates( S, NSTATES); /*
  numb
  ssSetNumDiscStates( S, 0); /* numb
  ssSetNumInputs( S, NINPUTS); /* numb
  ssSetNumOutputs( S, NOUTPUTS); /*
  numb
  ssSetDirectFeedThrough(S, 1); /* dire
  ssSetNumSampleTimes( S, 1); /* numb
  
```

Aero/Propulsion Model



Aero/Prop Model Block



Block name: aero/prop data	Done
Block type: S-Function	Revert
Subsystem: sys=fun(t,x,u,flag,param1,...)	Help
Subsystem function name: ux30dat2	
Function parameters: 	

```

c   perform table look-ups
c
c   call ux30d(mach,weight,alpha,delta,delta,
1  defl1,defl2,eta,specv25,altHo,vela,
      .
      .
c   Mass of vehicle
c
c   mass = weight/32.17
c
c   sum aero forces and moments
c
c   clift = clt + cldat + cldet
c   cdrag = cdt + cddat + cddet + cdf11 + cdf21
c
c   sinalf = sin(alpha*d2r)
c   cosalf = cos(alpha*d2r)
c
c   cx = -cdrag*cosalf + clift*sinalf
c   cy = cyb1*beta + cydat + cydet + cyf11 + cyf21
c   cz = -clift*cosalf - cdrag*sinalf
c
c   cil = cilb1*beta + cildat + cildet + cilf11 + cilf21
c   1 (clipt * pbody + clirt * rbody) * bapan/(2.0*vrw)
c   cm = cmat + cmdat + cmdet + cmf11 + cmf21 +
c   1 cmqt*qbody * cbar/(2.0*vrw)
c   cn = cwb1*beta + cwdat + cwdet + cwf11 + cwf21 +
c   1 (cwpt*pbody + cwrt * rbody) * bapan/(2.0*vrw)
c
c   final outputs from user code block
c
c   lift = dynp*sarea*clift
c   drag = dynp*sarea*cdrag
c   lovd = lift/drag
c
c   xzero = dynp*sarea*cx
c   yzero = dynp*sarea*cy
    
```

Linear Model Derivation

```
>> [ad,bd,cd,dd] = linmod('ux30');
>> ad
ad =
-2.2037e-02  6.9900e-03      0    -5.6189e-01  -8.4104e-03
-3.7593e-05  -9.7957e-02  1.0000e+00  -4.2937e-05  2.8210e-04
-3.7335e-02  3.8823e+00  -1.2216e-01      0    -3.8333e-04
 2.7302e-06  3.9183e-06  1.0000e+00  -3.9183e-06  1.6346e-14
 1.0472e-02  -1.3703e+02      0    1.3703e+02      0

>> bd
bd =
 4.5324e-02  4.1653e-01
-9.4348e-03  -2.1950e-03
-2.4606e+00  -2.4238e-03
      0      0
      0      0
```

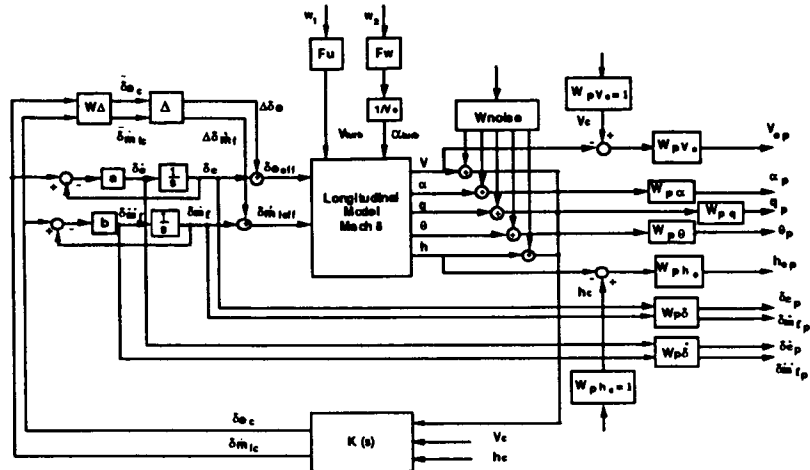
Linear Model Evaluation

```
>> eig(ad)

-2.0772e+00
 1.8636e+00
-5.4707e-02
 1.3076e-02 + 1.9857e-01i
 1.3076e-02 - 1.9857e-01i
```

- **Unstable system=> controller requirements**
 - Controller stabilize vehicle
 - Controller follows prescribed path

System Block Diagram



Controller Synthesis

```
>> Oiplant_UX1
system: 12 states 16 outputs 11 inputs
Test bounds: 0.5000 < gamma <= 10.0000
```

gamma	hamx_eig	xinf_eig	hamy_eig	yinf_eig	nrho_xy	p/f
10.000	3.9e-02	3.3e-10	9.6e-04	0.0e+00	0.0000	p
5.250	3.9e-02	3.3e-10	9.6e-04	0.0e+00	0.0000	p
						⋮
0.648	3.9e-02	-5.8e-05#	9.6e-04	0.0e+00	0.0003	f
0.723	3.9e-02	3.9e-10	9.6e-04	0.0e+00	0.0005	p
0.686	3.9e-02	-2.2e-02#	9.6e-04	0.0e+00	0.0419	f
0.704	3.9e-02	4.0e-10	9.6e-04	0.0e+00	0.0009	p
0.695	3.9e-02	4.0e-10	9.6e-04	0.0e+00	0.0017	p

Gamma value achieved: 0.6948

M-file Example

```

    ⋮
% Performance Weighting Functions on Actuators
Wpe = daug(20,5);      % weighting on dele, deleta
Wpact = daug(1,1);    % weighting on elevon rate
    ⋮
systemnames = 'ac Wpv Wph Wpa Wpq Wpo Wpe Wn se0 se1 se5 se6 Wpact
              Fv Fh Wcmdv Wcmdh ';

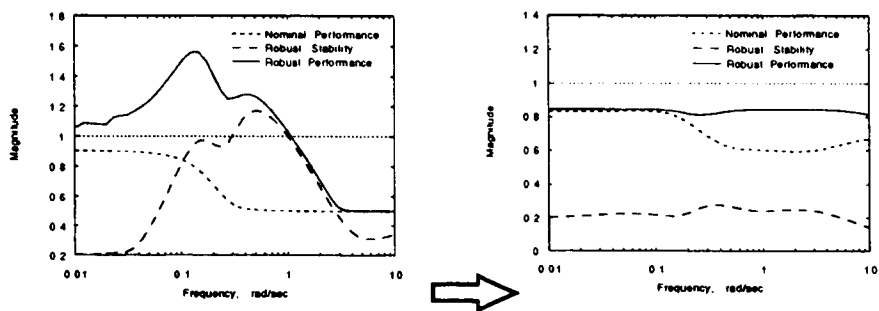
inputvar = '[xt_inp(11)];

outputvar='[Wpv;Wph;Wpa;Wpq;Wpo;Wpe;Wpact;ac+ Wn;Wcmdv;Wcmdh]';

input_to_ac='[Fh;Fv;se1;se6]';
input_to_Wn='[xt_inp(1:5)]';
input_to_Fh = '[xt_inp(6)]';
    ⋮
% H∞ controller calculation
[k1,clp1]=hinfyn(acolp1,nmeas,ncon,0.5,10,0.01);

```

Controller Evaluation Tools

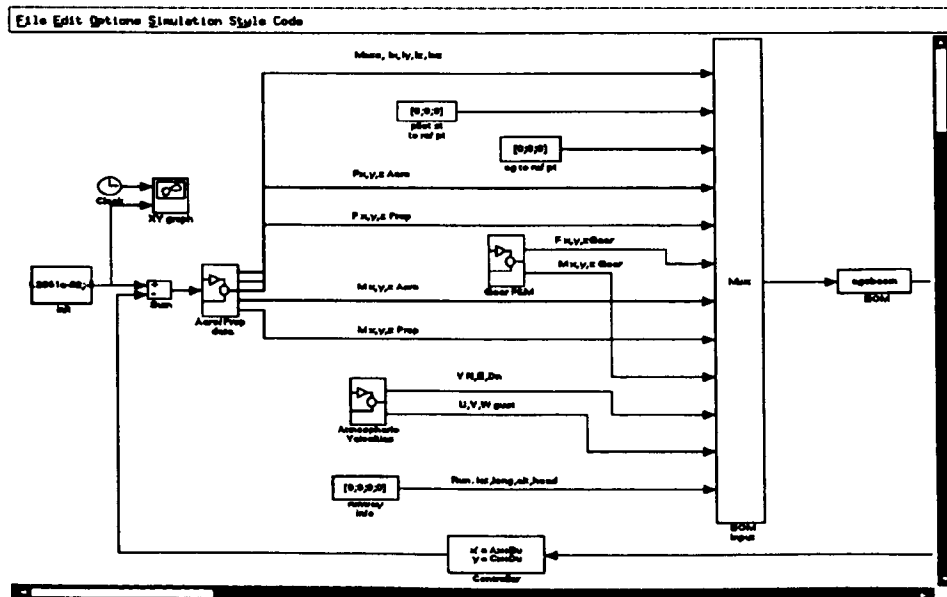


- Frequency domain μ -based performance evaluation plots

Controller Order Reduction

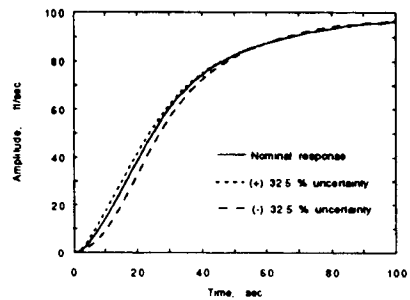
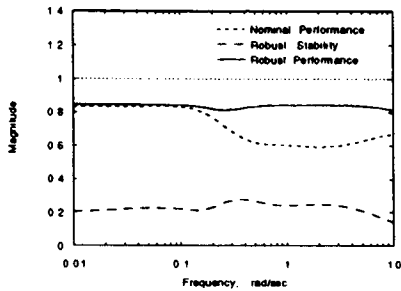
- Nominal controller - 23 states
- Balanced realization and Hankel singular values
 - >> [Kbal, hanksv] = sysbal(Khinf);
 - >> [Kred, Kunst] = hankmr(Kbal, hanksv, 13);
- Final reduced-order controller - 13 states

Nonlinear Simulation



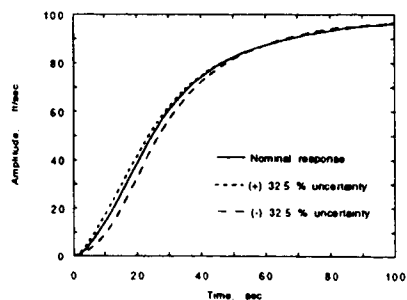
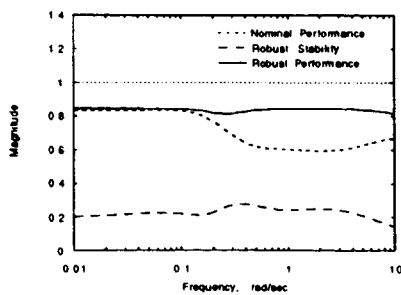
Controller Evaluation

- Reduced order controller evaluation in frequency and time domain
 - μ (closed loop system)
 - nonlinear simulation time response



Controller Evaluation

- Reduced order controller evaluation in frequency and time domain



Concluding Remarks

- **Matlab capabilities utilized**
 - Link together FORTRAN, C, and Matlab functions
 - Nonlinear simulation
 - Trim vehicle
 - Derive linear model
 - Control application toolboxes for controller synthesis and analysis

356143

110060

N95-16475

P. 22

Simulation of the Coupled Multi-Spacecraft Control Testbed at the Marshall Space Flight Center

D. Ghosh and R.C. Montgomery
NASA Langley Research Center, Hampton VA 23681

1994 NASA Langley Workshop on Software Systems
June 15-16, 1994
Hampton, VA

The capture and berthing of a controlled spacecraft using a robotic manipulator is an important technology for future space missions and is presently being considered as a backup option for direct docking of the Space Shuttle to the Space Station during assembly missions. The dynamics and control of spacecraft configurations that are manipulator-coupled with each spacecraft having independent attitude control systems is not well understood and NASA is actively involved in both analytic research on this three-dimensional control problem for manipulator-coupled active spacecraft and experimental research using a two-dimensional ground based facility at the Marshall Space Flight Center (MSFC). This paper first describes the MSFC testbed and then describes a two-link arm simulator that has been developed to facilitate control theory development and test planning. The motion of the arms and the payload is controlled by motors located at the shoulder, elbow and wrist.

A symbolic manipulator, MAPLE, is used to derive the equations of motion based on a Lagrangian formulation. The equations are programmed using the autocode feature of MAPLE in FORTRAN and are then embedded in a usercode block of MatrixX which is the primary simulation software engine. The simulator implements a digital joint motor controller. The joint motor control scheme generates commands for the motor based on the difference between the joint angles derived from telerobotic translational command inputs using inverse kinematics and joint angle measurements.

Simulation of the Coupled Multi-Spacecraft Control Testbed at the Marshall Space Flight Center

D. Ghosh and R. C. Montgomery
NASA Langley Research Center
Hampton, VA 23681

1994 NASA Langley Workshop on Software Systems

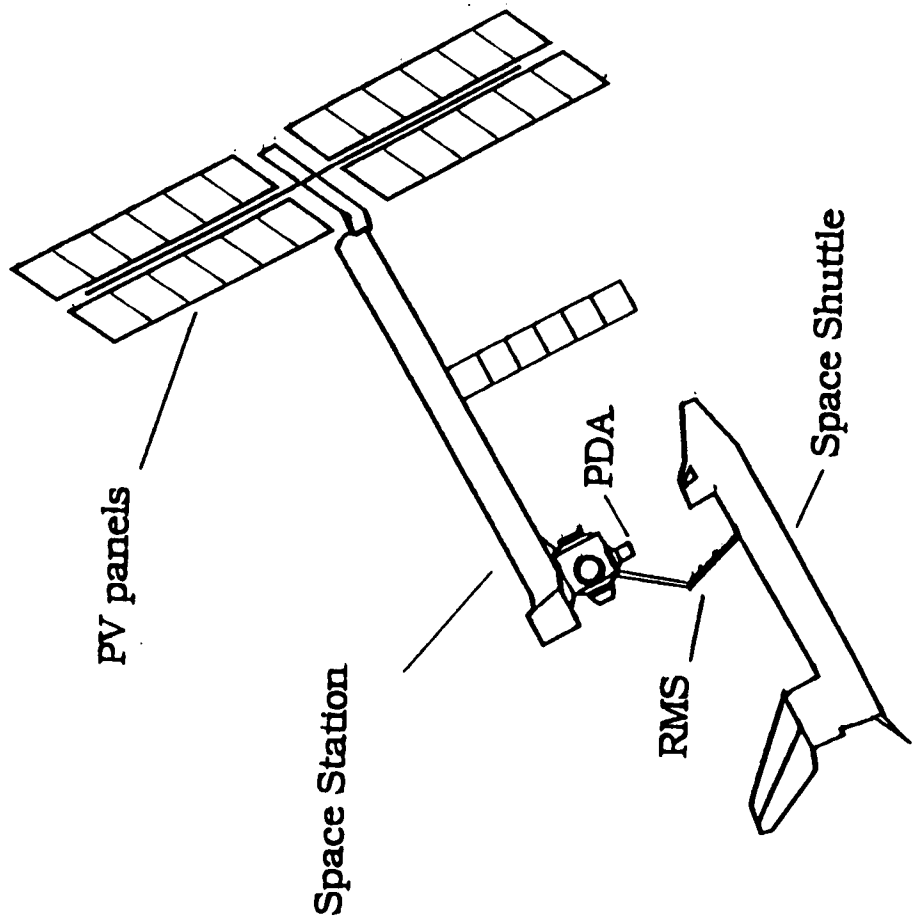
June 15 -16, 1994
Hampton, Va

PRESENTATION OUTLINE

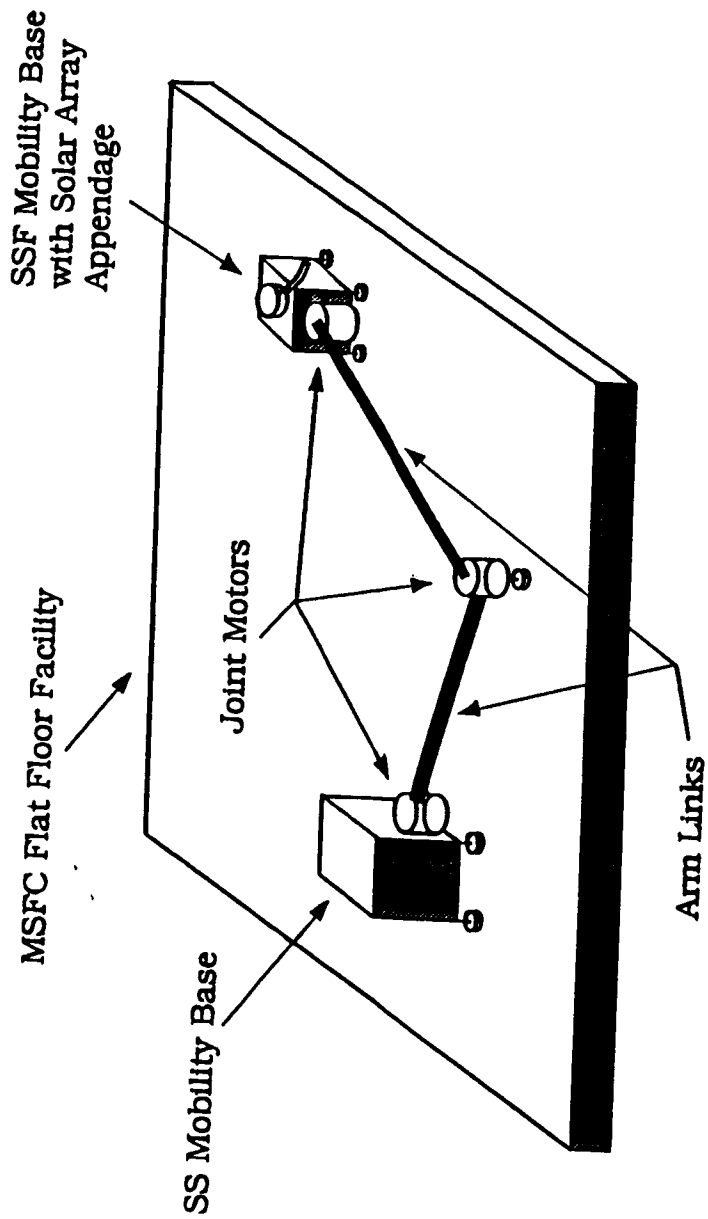
- Problem
- Research Facility
- Simulator
 - Overview
 - Modelling
 - System
- Results
- Concluding Remarks

Multi-Body Spacecraft Control Problem

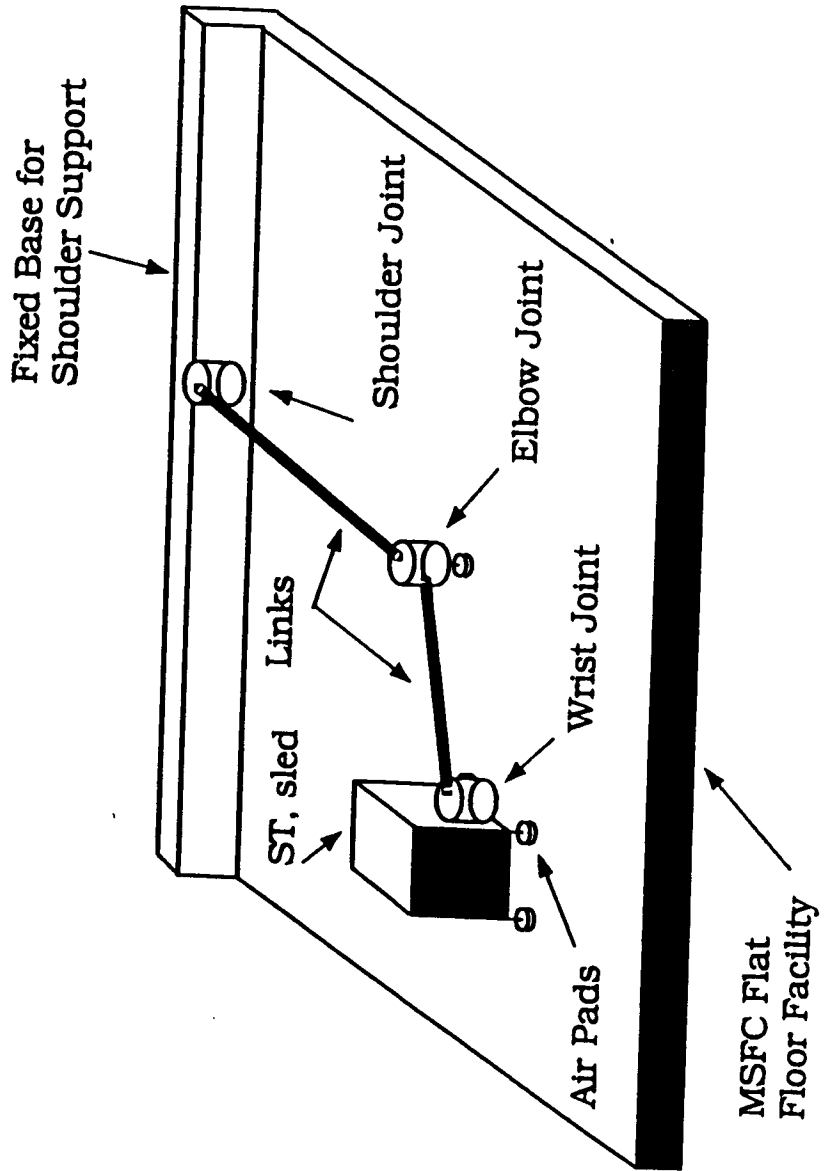
Space Station Berthing to the Space Shuttle



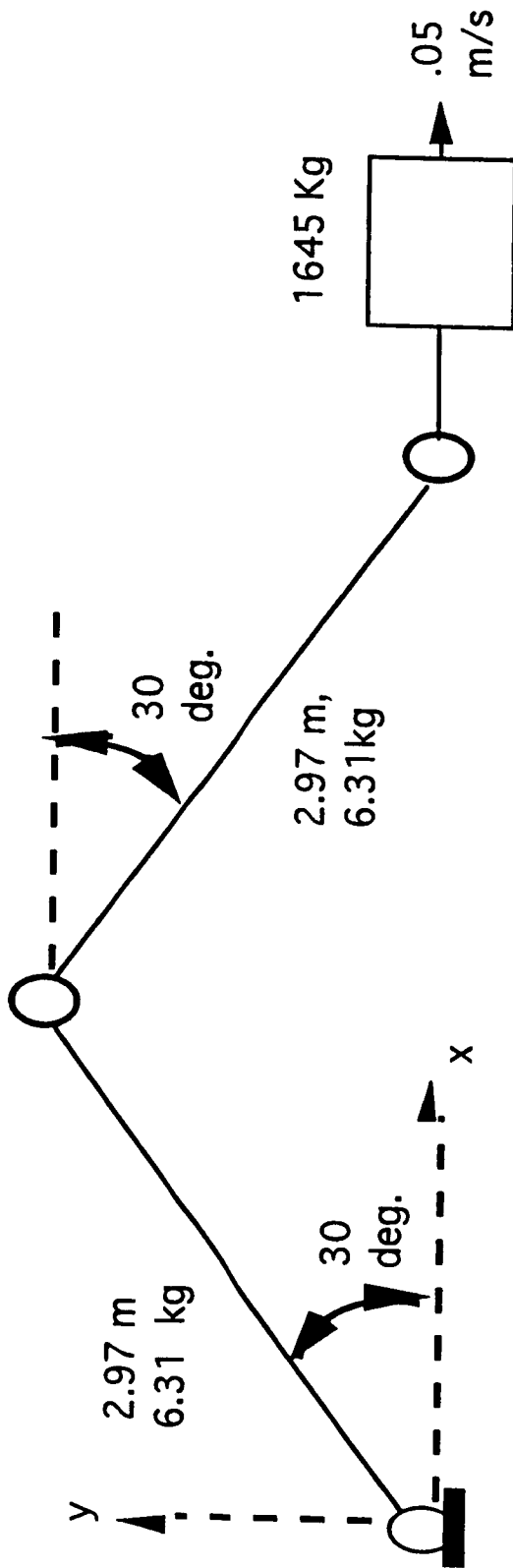
PLANNED RESEARCH TESTBED



CURRENT RESEARCH TESTBED



Physical Parameters



SIMULATOR

Overview

- **Derive Equations of Motion (EOM) - MAPLE**
- **Numerically integrate EOM for a given input - MatrixX**

SIMULATOR

Derivation of Equations of Motion

- **Based on Lagrangian Formulation**
- **Employs Symbolic Manipulation (MAPLE)**
- **Code for the Equations of Motion are generated in FORTRAN**
- **Equations are embedded in a usercode block of MatrixX**

MODELLING

Equations of Motion

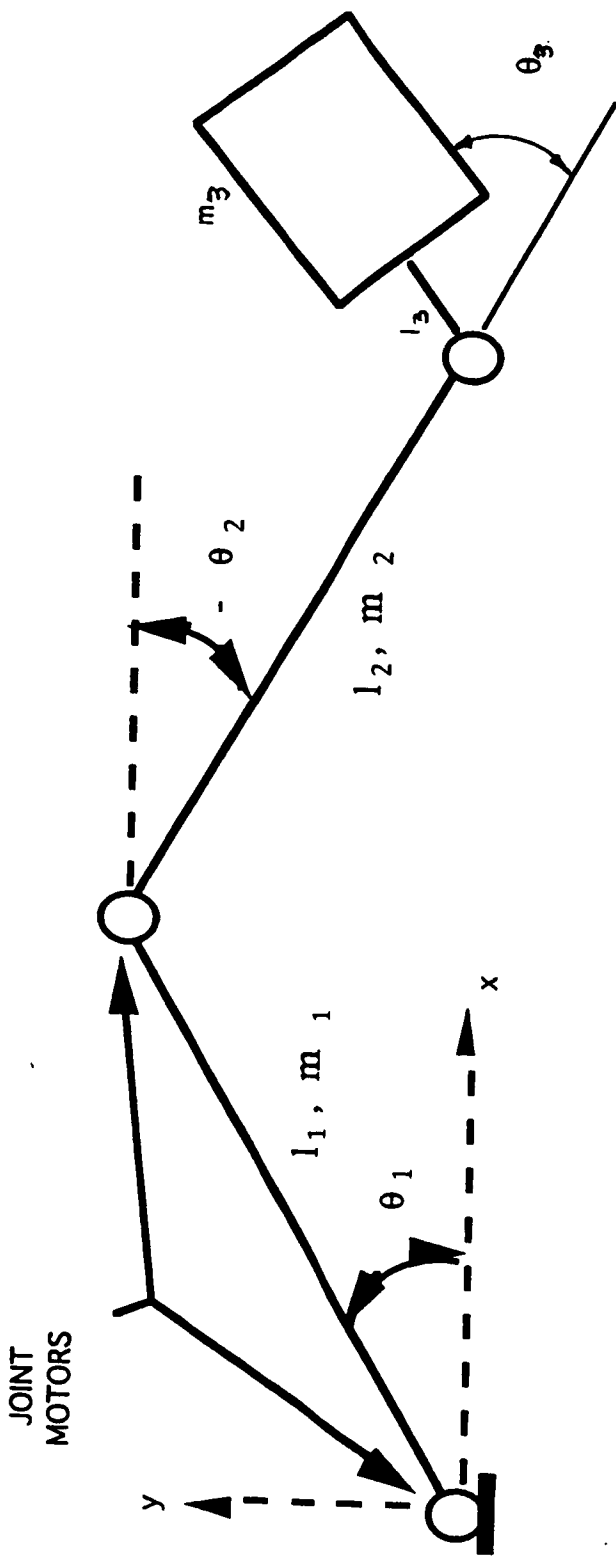
$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \frac{\partial W}{\partial q_i} - \frac{\partial F}{\partial \dot{q}_i}$$

Lagrangian : $L = T - V$

Raleigh Dissipation : F

Virtual Work : W

System Model Used



MODELLING

(continued)

• Kinetic Energy

$$T = \frac{1}{2} \sum_{i=1}^3 I_{S_i} \dot{\theta}_i^2 + \frac{1}{2} \frac{m_1 l_1^2}{3} \dot{\theta}_1^2 +$$

$$\frac{m_2}{2} [(l_1 \dot{\theta}_1 \cos \theta_1 + \frac{l_2}{2} \dot{\theta}_2 \cos \theta_2)^2$$

$$+ (l_1 \dot{\theta}_1 \sin \theta_1 + \frac{l_2}{2} \dot{\theta}_2 \sin \theta_2)^2] + \frac{1}{2} \frac{m_2 l_2^2}{12} \dot{\theta}_2^2$$

$$+ \frac{m_3}{2} [(l_1 \dot{\theta}_1 \cos \theta_1 + l_2 \dot{\theta}_2 \cos \theta_2 + l_3 \dot{\theta}_3 \cos \theta_3)^2$$

$$+ (l_1 \dot{\theta}_1 \sin \theta_1 + l_2 \dot{\theta}_2 \sin \theta_2 + l_3 \dot{\theta}_3 \sin \theta_3)^2]$$

MODELLING

(Continued)

Potential Energy

$$V = \frac{1}{2} k_{x1} (\theta_1 - \theta_{s1})^2 + \frac{1}{2} k_{x2} (\theta_2 - \theta_{s2})^2 + \frac{1}{2} k_{x3} (\theta_3 - \theta_{s3})^2$$

Virtual Work $W = T_{E1} \theta_{s1} + T_{E2} (\theta_{s2} - \theta_1) + T_{E3} (\theta_{s3} - \theta_2)$

Raleigh dissipation

$$F = \frac{1}{2} k_{v1} (\dot{q}_1 - \dot{q}_{s1})^2 + \frac{1}{2} k_{v2} (\dot{q}_2 - \dot{q}_{s2})^2 + \frac{1}{2} k_{v3} (\dot{q}_3 - \dot{q}_{s3})^2$$

EQUATIONS OF MOTION

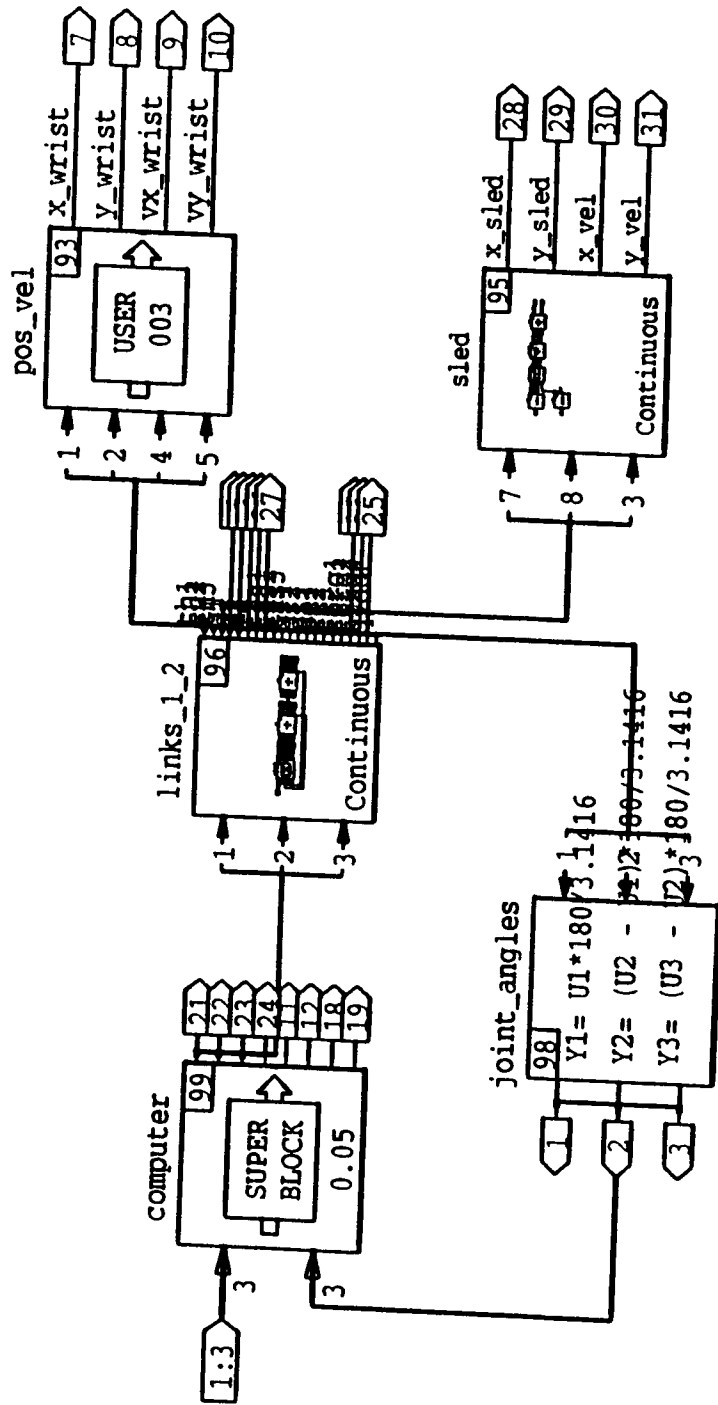
- Coupled non-linear ODEs

$$\mathbf{M}(\theta)\ddot{\theta} + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} + \mathbf{K}\theta = \mathbf{Q}$$

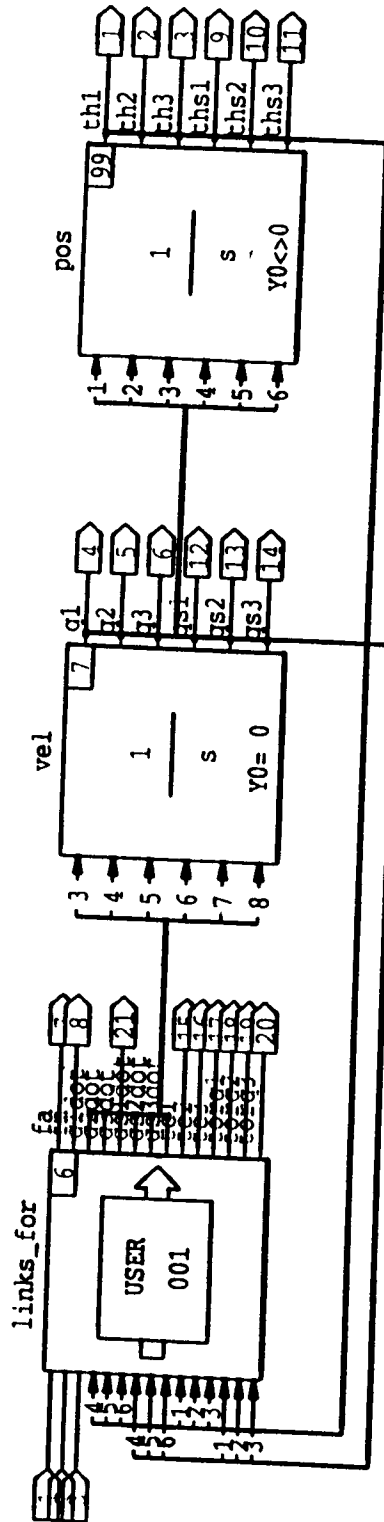
$$\ddot{\theta} = \mathbf{M}^{-1}(\theta) (\mathbf{Q} - \mathbf{C}(\theta, \dot{\theta})\dot{\theta} - \mathbf{K}\theta)$$

Matrix inversion done symbolically by MAPLE

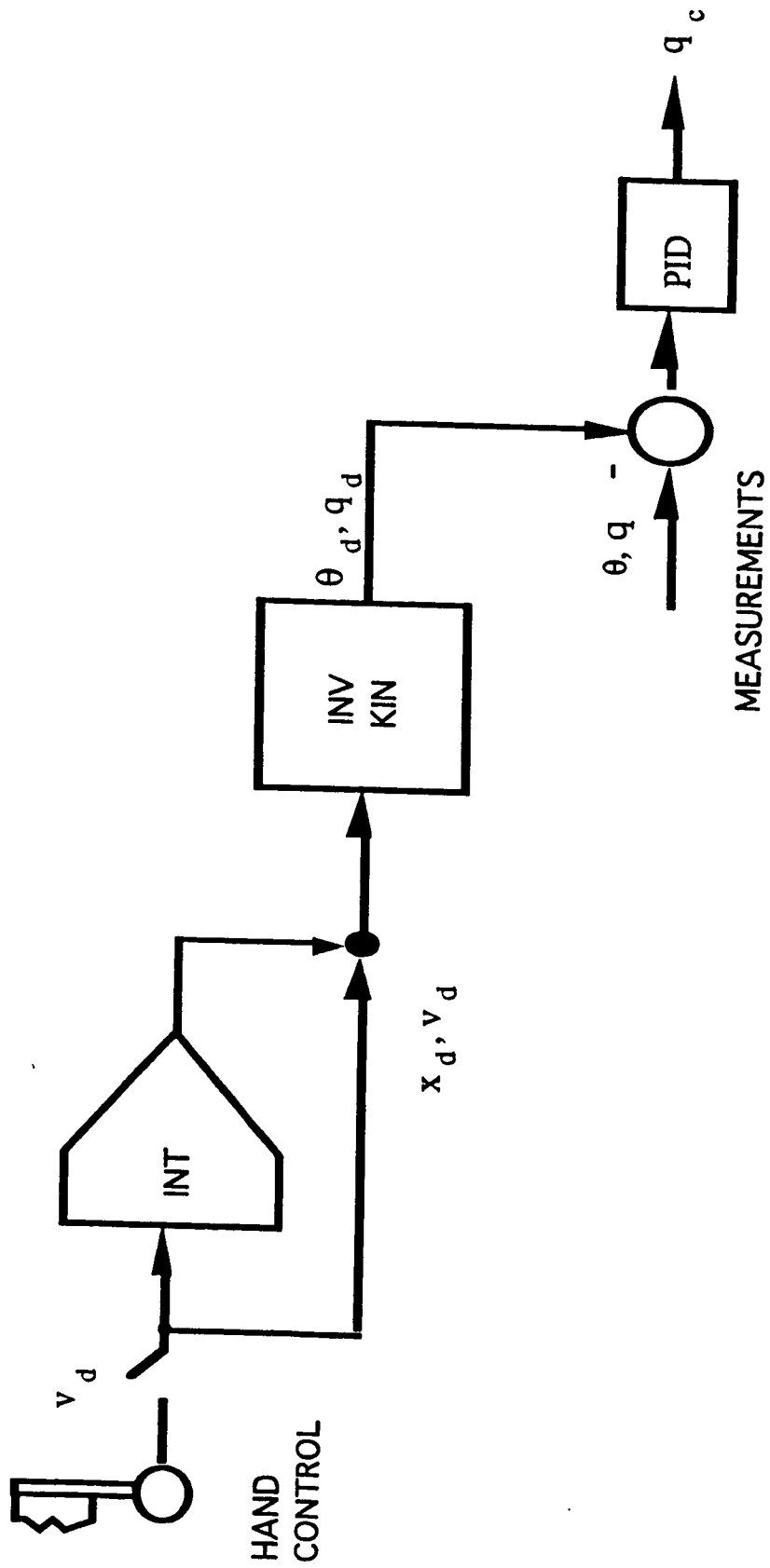
System Simulator



Subsystem

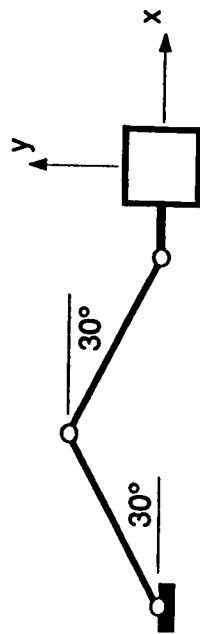


Control System

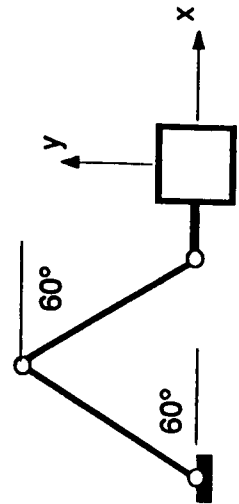


TEST MANEUVER

INITIAL CONFIGURATION



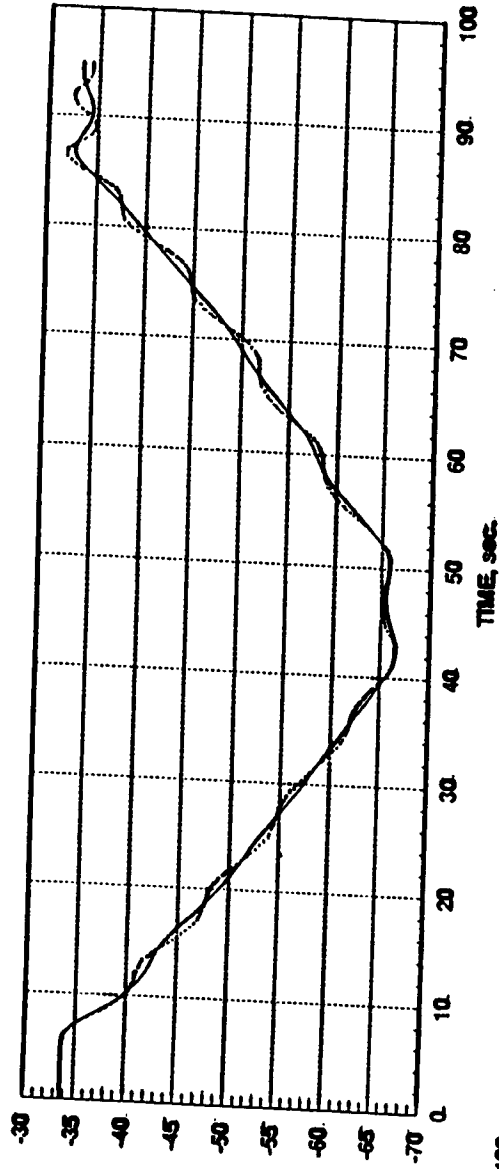
RETRACTED CONFIGURATION



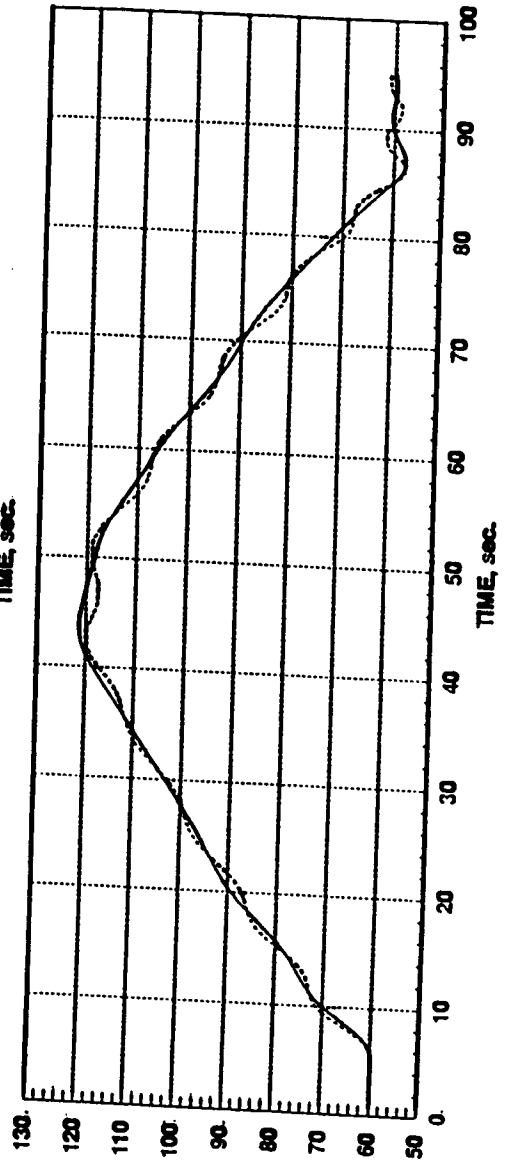
RESULTS

Payload x-velocity = .061 m/s

Shoulder joint angle, deg.

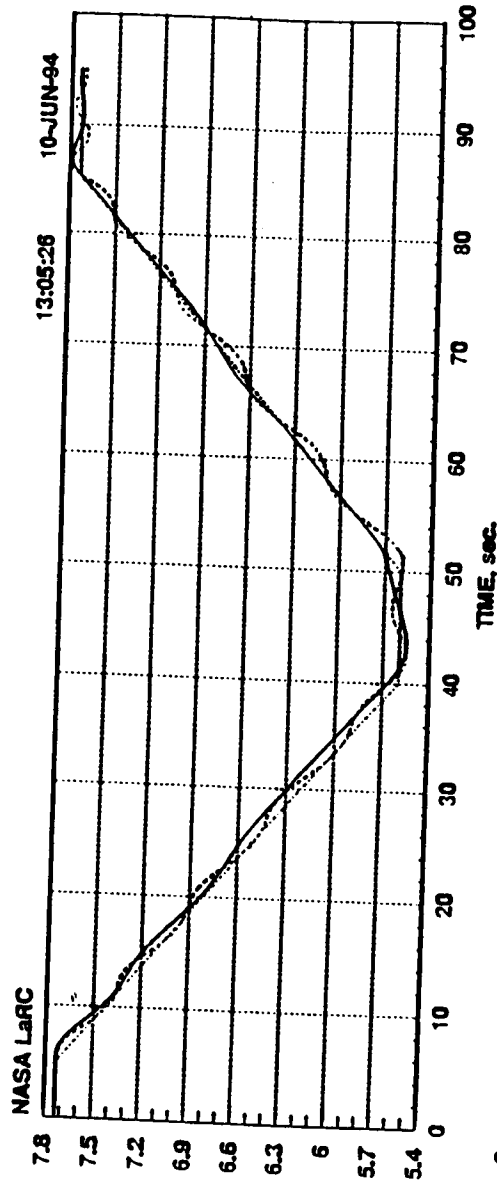


Elbow joint angle, deg.

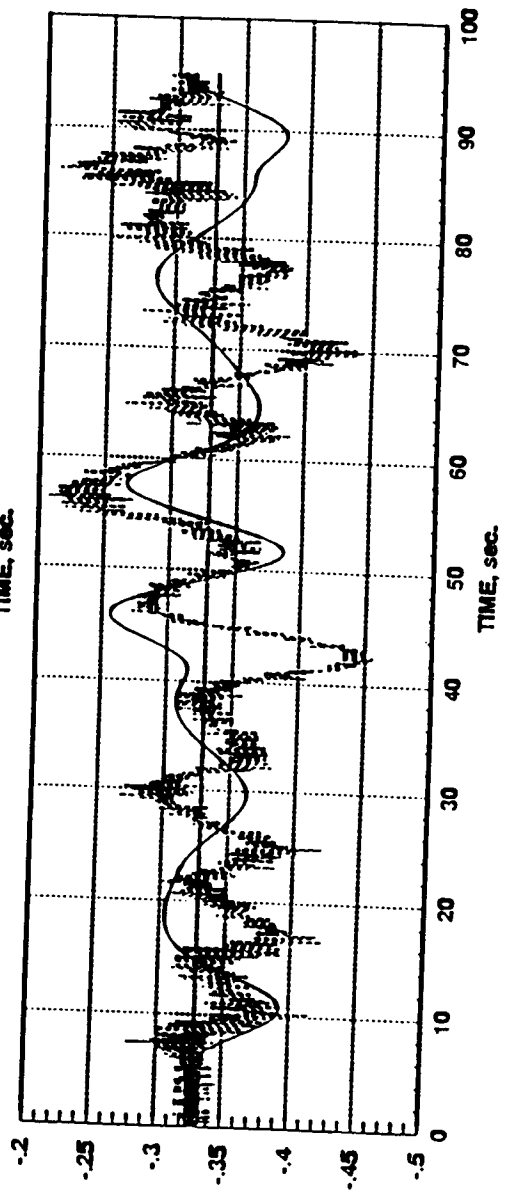


RESULTS (cont)

Payload x-velocity = .061 m/s



X disp, m



Y disp, m

Concluding Remarks

- **Results are encouraging**
- **Simulation tools used effectively**
- **Improvements needed in modelling**

SESSION 10 Languages

Chaired by

Robert F. Estes

10.1 Object Oriented Numerical Computing in C++ - John Van Rosendale

10.2 Hardware Description Languages - Jerry H. Tucker

10.3 High Performance FORTRAN - Piyush Mehrotra

1994 Workshop on The Role of Computers in LaRC R&D

Object Oriented Numerical Computing in C++

John Van Rosendale

Institute for Computer Applications in Science and Engineering
jvr@icase.edu

P. 17

Synopsis

C++ is an efficient object-oriented language of rapidly growing popularity. It can be of real value in a wide range of disciplines, including numerical computing, where it seems to offer important advantages over most competing languages.

Object-oriented languages

What exactly is an object-oriented language? The most important defining characteristic is support for "polymorphic data types." Procedural languages, like Fortran and C, contain built-in types such as integers, reals, characters and so on. The integer type, for example, consists of the requisite bits of data, a set of associated operations, +, *, /, ..., and coercions to and from the other built-in types. One can build data structures of arbitrary complexity in Fortran, but these are not "first class" types, like integers.

For example, one can form a "sparse_matrix" from arrays of integers indexing into arrays of reals. But Fortran 77 does not let one declare several of these as

```
sparse_matrix A,B,C
```

and then perform operations such as:

```
A = B + C
```

Languages like Clu and Ada, supporting "abstract data types," let one do precisely this. One can, for example, in Ada define a "set_of_words" abstract data type. This would be a user defined type which might be useful in comparing documents. Once the type is defined, one can then declare several such sets

```
set_of_words A,B,C
```

One can also operate on them just as with the built in types

```
A := B .+ C
```

where .+ might be a user-defined union operation.

OO languages push this concept further, allowing one to define a "set_of_<type T>", where T can be any type in the language. This new type, a "set_of_<type T>", is "first class" in OO languages, one can use variables of that type exactly like those of the built-in types. To make this clear, types are called "classes" in the OO world, while values of those types (classes) are called "objects," though whether these new terms do more to clarify or obfuscate is not clear.

To see how OO ideas might be used in numerical computing, it might, for example, be useful to define a class "mesh_cell" which would be the basic unit of an unstructured mesh. Mesh cells come in a number of varieties, which can be thought of as subtypes (subclasses) of the type (class) "mesh_cell", as shown in Figure .

All mesh cells share certain properties, volume, temperature, pressure etc. declared as part of class "mesh_cell." Cubes and tetrahedrons share these properties, but have their own unique properties as well. They have different numbers of faces and vertices for example.

The ability to allow useful computing on a set of related but not identical user-defined types is the defining characteristic of OO languages. In the above case, one can make an array of "mesh_cells", consisting of prisms, tetrahedrons, and cubes. One can access the volume of any element of this array, since all "mesh_cells" have volume. To access specialized properties, one may have to select on the particular subclass of each "mesh_cell".

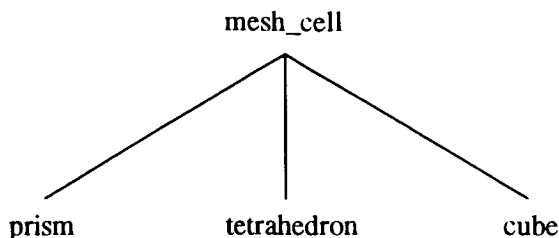


Figure 1: Mesh cell type hierarchy

C++ in numerical computing

How useful will C++ be in numerical computing? C++ contains most of the useful new features in Ada or Fortran 90, and is easily extensible in a number of ways. People around the world are rapidly developing class libraries for finite element analysis, for sparse matrix arithmetic, and so on. C++ together with a new class library is essentially a new application-specific language, and one that may have a powerful impact on a particular subdiscipline.

To see how this could have an impact, one need only realize that there are, for example, at least a dozen different unstructured grid codes here at Langley, with relatively little code shared between them. Given the appropriate class library supporting unstructured grids, one should be able to prototype new unstructured grid algorithms much faster, by borrowing large chunks of previously written code. This is the promise of OO computing in C++. Efficient execution, compatibility with previously written C and Fortran, and the OO approach are the major advantages to C++.

C++ also has its problems. One is that its syntax and semantics, inherited from C, are needlessly complex, significantly steeping the learning curve for new programmers. Another problem is that, like Ada and Fortran 90, C++ is a large language, full of complexities most programmers will never master. Only experts will master the full language, with most programmers limping along on their own particular subset.

These problems are real, but clearly not fatal, given the exponential growth of C++. From one perspective, C++ is essentially a halfway point between traditional procedural languages, like C and Fortran, and "rapid prototyping" languages like Smalltalk. Over the longer term, as computer power increases and our algorithms become more complex, one expects research numerical computing, like that done at Langley, to shift in the "rapid prototyping" direction. Use of C++ is an important step in that direction.

Object Oriented Numerical Computing in C++

1994 Workshop on The Role of Computers in LaRC R&D

John Van Rosendale

jvr@icase.edu

Institute for Computer Applications in Science and Engineering

An **object oriented** language is one allowing users to create a set of related types and then intermix and manipulate values of these related types.

- Such types are called **classes**.
- Values of such types are called **objects**.

Intermixing related user-defined types is called **polymorphism**. Polymorphism, and the things that go with it, *encapsulation*, *inheritance*, and *dynamic binding* give OO languages their increased semantic power.

Some Well Known OO Languages

Smalltalk

C++

Owl/Trellis

Eiffel

Objective C

Objective Pascal

Actors

Loops

Clos

Simula

Fortran is not OO. Integer variables, for example, have powerful properties the user cannot duplicate in new types. There is no way to define a sparse-matrix type, then do:

```
sparse_matrix A,B,C
```

```
A = B + C
```

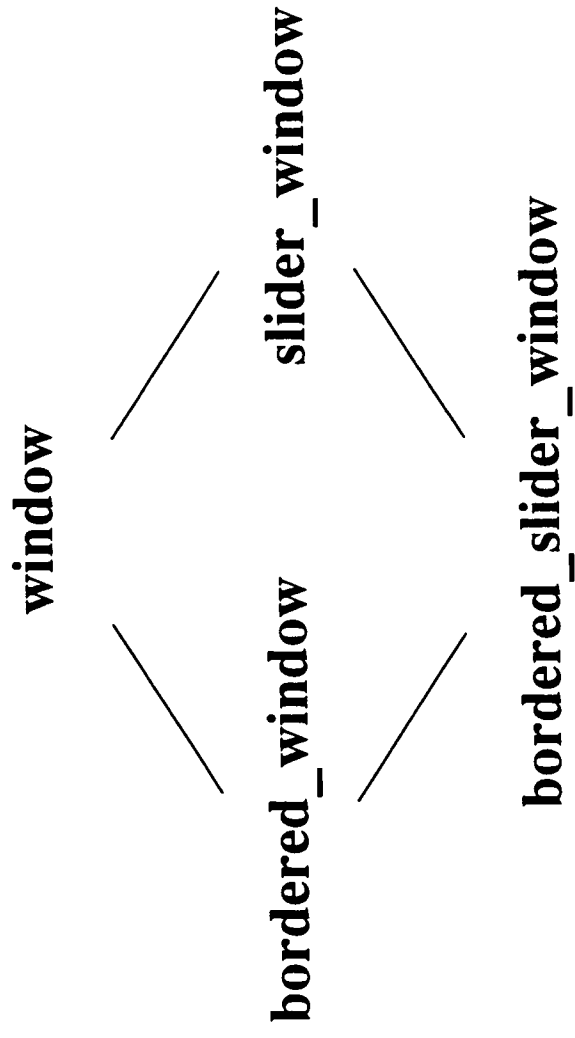
Languages supporting “abstract data types” allow users to define new types. In Ada, one can create a “sparse_matrix” type or a “set_of_words” type:

```
set_of_words A,B,C
```

```
A := B .+. C
```

where `.+.` is a user defined union operator.

Typical class lattice:



What is C++ ?

- A statically-checked object oriented language downward compatible with C
- Useful on any program with complex data structures (though originally intended for systems programming)
- Potentially useful in scientific programming, since it is efficient and supports complex data structures well

Templates – data abstractions with types as parameters

```
template<class T>
class stack {
    T* v;
    T* p;
    int sz;

    public:
        stack (int s){ y = p = new T[sz = s]; }
        ~stack () { delete[] v; }

        void push(T a){ *p++=a; }
        T pop(){ return *--p; }

        int size() const { return p-v; }
};
```

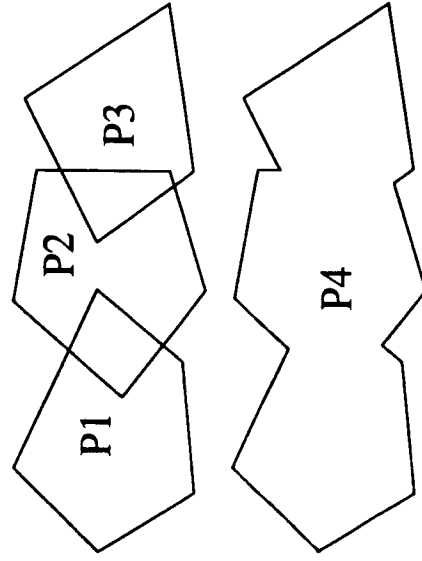
Operator Overloading – the ability to define new uses for the built-in operator symbols (+ - / ! = ...)

Example: polygon union

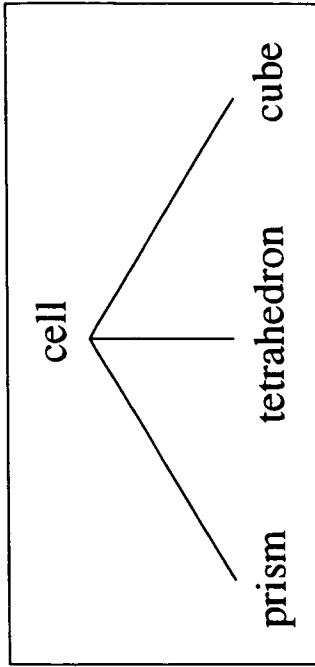
$P4 = P1 + P2 + P3$

$P4 = \text{union}(\text{union}(P1, P2), P3)$

$P4 = P1.\text{union}(P2.\text{union}(P3))$



Using inheritance in numerical codes



cell:

- volume
- flow variables

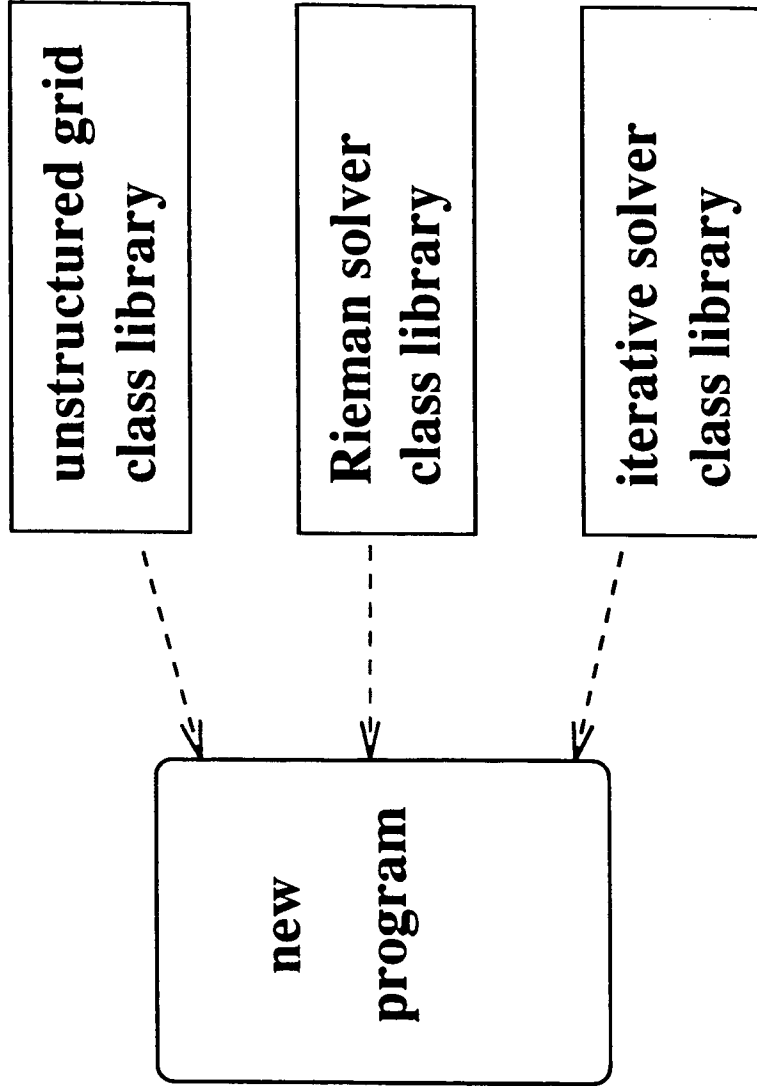
cube:

- 8 vertices
- 6 faces

prism:

- 6 vertices
- 5 faces

Potential for code reuse:



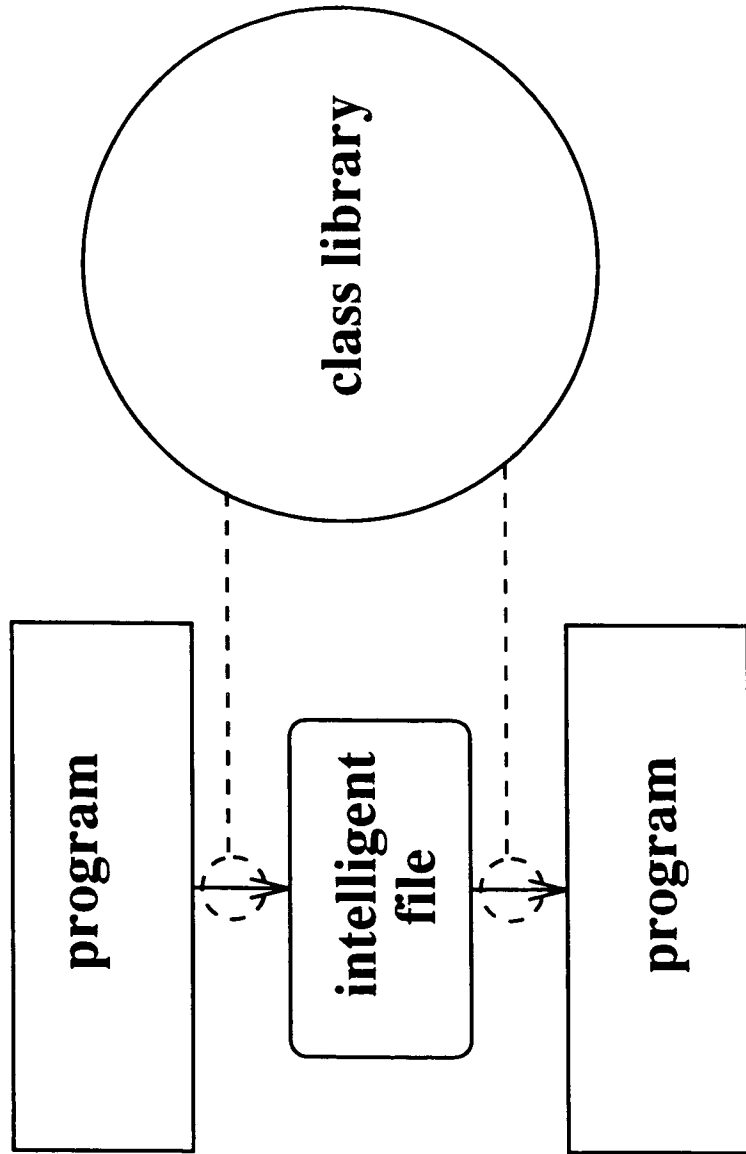
Efficiency

- C++ provides much of the semantic power of dynamically typed languages, like Smalltalk, but is much more efficient.
- C++ code runs about as fast as C or Fortran, if one avoids polymorphism and abstractions.
- The overhead in using large objects (e.g. a sparse matrix object) is minor.
- Fine grained objects, such as complex variables, are inefficient and should usually be avoided.

Observations

- Efficiency should be thought of in terms of the entire programming, debugging, and execution cycle.
- If a language would make programming substantially easier, there could be such a gain from algorithmic improvements, that significant run-time inefficiency could be tolerated.

Interoperability



Conclusions

- C++ is an effective OO language, and has become the defacto standard.
- C++ supports the data structures needed for complex numerical algorithms very well.
- It is efficient, and can be readily intermixed with C, Fortran and perhaps HPF.
- Developing numerical algorithms in C++ should increase opportunities for code reuse and for sharing code between programmers.

Hardware Description Languages

P. 10

Jerry H. Tucker

Hardware description languages are special purpose programming languages. They are primarily used to specify the behavior of digital systems, and are rapidly replacing traditional digital system design techniques. This is because they allow the designer to concentrate on how the system should operate rather than on implementation details. Hardware description languages allow a digital system to be described with a wide range of abstraction, and they support top down design techniques. A key feature of any hardware description language environment is its ability to simulate the modeled system.

The two most important hardware description languages are Verilog and VHDL. Verilog has been the dominant language for the design of application specific integrated circuits (ASIC's); however, VHDL is rapidly gaining in popularity. VHDL was developed for the DOD and then transferred to the IEEE in 1986. The language is defined by IEEE standard 1076. Since 1988 the DOD has required all of its digital ASIC's to be supplied with VHDL descriptions.

By describing a digital system in VHDL at a behavioral level, the effect of different architectural decisions can be simulated and evaluated early in the design process. Once an architecture has been selected the various circuits in that architecture can be described using a restricted subset of VHDL. It is then possible to synthesize that VHDL description to obtain the actual implementation of the circuit.

Hardware Description Languages

by

Jerry H. Tucker

Presented at the Workshop on
The Role of Computers in LaRC R&D
June 15-16, 1994

Questions Addressed

- What are HDL's?
- Why use HDL's?
- What HDL's are available?
- How do HDL's differ from other languages?

What are HDL's?

- A special purpose programming language.
- Primarily for specifying behavior and structure of digital systems.
 - Replaces traditional digital design techniques.
 - Supports wide range of system abstraction.
 - Supports top down design.
- Running the HDL program simulates the modeled system.

Why use HDL's?

- Old design methods are inadequate to satisfy demands on digital systems.
 - Increasing complexity.
 - Decreasing development time.
- Automates design process.
 - Requires digital hardware designers to also be programmers.
- HDL synthesized to implement design.

Types of HDL's

- Two dominant HDL's.
- Verilog
- VHDL
- A key component of both is the simulator.

Verilog

- Developed 1983-1984.
- Originally proprietary.
- Now IEEE 1364.
- Dominant language for ASIC's.
 - More “real” designs in Verilog.
- Inherently faster simulation than VHDL.

VHDL

- DOD required common HDL to support designs from different vendors.
- DOD contract awarded in 1983.
- Strong Ada influence.
- Public released 1985.

VHDL (cont.)

- Transferred to IEEE in 1986.
- IEEE standard 1076 in 1987.
- Revised standard IEEE 1076-1993.
- Since 1988 DOD requires all its digital ASIC's to be supplied with VHDL descriptions.

VHDL (cont.)

- VHDL more verbose than Verilog
- Example in VHDL
IF ((clk'EVENT) and (clk='1') and
(clk'LAST_VALUE='0')) then ...
- Example in Verilog
@(posedge(clk)) ...

VHDL (cont.)

- VHDL more flexible than Verilog.
- Momentum seems to be with VHDL.

Levels of Design

- Behavioral
 - Highest level, Most general.
- Register Transfer Level (RTL)
 - Defines registers, counters, I/O buffers etc.
 - Can be synthesized to specific devices.
- Gate Level
 - Defines design in terms of logic primitives.

VHDL Example mod 3 counter

```
-- MOD 3 counter VHDL example for
-- The role of computers in LaRC R&D
-- workshop June 15-16, 1994.
use work.all;
entity CNT is
  port(CLK: in BIT; Q1, Q0: out BIT);
end CNT;
```

mod 3 counter (cont.)

```
architecture BEHAVIOR of CNT is
begin
  CNT3: process(CLK)
    variable COUNT: INTEGER := 0;
  begin
    if CLK = '1' then
      COUNT := COUNT + 1;
```

mod 3 counter (cont.)

```
if (COUNT > 3) then
  COUNT := 0;
end if;
Q0 <= bit'val(COUNT mod 2) after 10 ns;
Q1 <= bit'val(COUNT/2) after 10 ns;
end if;
end process CNT3;
end BEHAVIOR;
```


VHDL example test bench

```
-- Test bench for MOD 3 counter VHDL example for
-- The role of computers in LaRC R&D
-- workshop June 15-16, 1994.
use work.all;
```

```
entity TB is end TB;
```

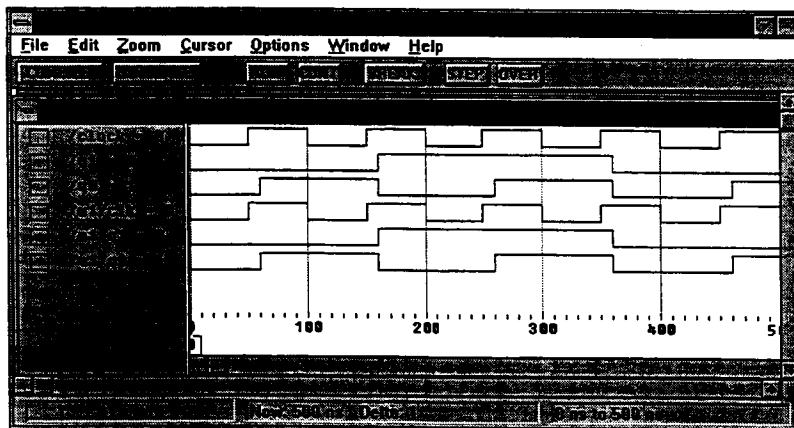
test bench (cont.)

```
architecture TEST of TB is
-- Signal declaration.
  signal CLOCK, Q1, Q0: BIT;
-- Component declaration.
  component CNT
    port(CLK: in BIT; Q1, Q0: out BIT);
  end component;
```

test bench (cont.)

```
for U1: CNT use entity work.CNT(BEHAVIOR);  
begin  
-- component instantiation statement.  
  U1: CNT port map(CLOCK, Q1, Q0);  
  
  CLOCK <= not CLOCK after 50 ns;  
end test;
```

Simulation of example



1994 Workshop on The Role of Computers in LaRC R&D

High Performance Fortran P. 17

Piyush Mehrotra

Institute for Computer Applications in Science and Engineering
pm@icase.edu

Introduction

Recently an international group of researchers from academia, industry and government labs formed the High Performance Fortran Forum aimed at providing an intermediate approach in which the user and the compiler share responsibility for exploiting parallelism. The main goal of the group has been to design a high-level set of standard extensions to Fortran called, High Performance Fortran (HPF), intended to exploit a wide variety of parallel architectures [2, 4].

A major performance issue of most parallel machines including distributed memory machines and non-uniform access shared memory machines, is the locality of data. The HPF extensions allow the user to carefully control the distribution of data across the memories of the target machine. However, the computation code itself is written using a global name space independent of the distribution of the data. As HPF is targeted towards data parallel algorithms, it supports forall loops and array statements to specify the data parallelism. However, there are no explicit constructs for management of tasks or for communication of data. It is the compiler's responsibility to analyze the distribution annotations and generate parallel code, generally SPMD, inserting synchronization where required by the computation. Thus, using this approach the programmer can focus on high-level algorithmic and performance critical issues such as load balance while allowing the compiler system to deal with the complex low-level machine specific details.

The HPF design is based on language research done by several groups, in particular, Kali [5, 6], Vienna Fortran [1, 7] and Fortran D [3], the first two of these were partially developed at ICASE.

HPF Overview

High Performance Fortran is a set of extensions for Fortran 90 designed to allow specification of data parallel algorithms. The programmer annotates the program with distribution directives to specify the desired layout of data. The underlying programming model provides a global name space and a single thread of control. Explicitly parallel constructs allow the expression of fairly controlled forms of parallelism, in particular data parallelism. Thus, the code is specified in high level portable manner with no explicit tasking or communication statements. The goal is to allow architecture specific compilers to generate efficient code for a wide variety of architectures including SIMD, MIMD shared and distributed memory machines.

Fortran 90 was used as a base for HPF extensions for two reasons. First, a large percentage of scientific codes are still written in Fortran (Fortran 77 that is) providing programmers using HPF with a familiar base. Second, the array operations as defined for Fortran 90 make it eminently suitable for data parallel algorithms.

Features of High Performance Fortran

In this subsection we provide a brief overview of the new features defined by HPF.

- *Data mapping directives:* HPF provides an extensive set of directives to specify the distribution and alignment of arrays. The distribution directives can be used to specify the layout of data arrays on an underlying set of abstract processors. The alignment directives allow the arrays to be aligned so that specified elements are placed on the same abstract processors.

- *Data parallel execution features:* The **FORALL** statement and construct and the **INDEPENDENT** directive can be used to specify data parallel code. The concept of *pure* procedures callable from parallel constructs has also been defined.
- *New intrinsic and library functions:* HPF provides a set of new intrinsic functions including system functions to inquire about the underlying hardware, mapping inquiry functions to inquire about the distribution of the data structures and a few computational intrinsic functions. A set of new library routines have also been defined so as to provide a standard interface for highly useful parallel operations such as reduction functions, combining scatter functions, prefix and suffix functions, and sorting functions.
- *Extrinsic procedures:* HPF is well suited for data parallel programming. However, in order to accommodate other programming paradigms, HPF provides *extrinsic* procedures. These define an explicit interface and allow codes expressed using a different paradigm, such as an explicit message passing routine, to be called from an HPF program.

Full details of the language can be found in the HPF Language Specification document [2] which is also available via anonymous ftp from public/HPFF/draft at titan.cs.rice.edu.

There is a second round of meetings of the *High Performance Fortran Forum* being currently held in Chicago to consider clarifications of HPF 1 and to chart out requirements for future extensions to HPF. Further information about these meetings and HPF in general can be found on Mosaic through the URL <http://www.erc.msstate.edu/hpff/home.html>

References

- [1] B. Chapman, P. Mehrotra, and H. Zima. Programming in Vienna Fortran. *Scientific Programming*, 1(1):31-50, 1992.
- [2] High Performance Fortran Forum. High Performance Fortran Language Specification Version 1.0. *Scientific Programming*, 2((1-2)):1-170, Spring and Summer 1993.
- [3] G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, C. Tseng, and M. Wu. Fortran D language specification. Department of Computer Science Rice COMP TR90079, Rice University, March 1991.
- [4] C. Koelbel, D. Loveman, R Schreiber, G. Steele, and M. Zosel. *The High Performance Fortran Handbook*. The MIT Press, 1994.
- [5] P. Mehrotra. Programming parallel architectures: The BLAZE family of languages. In *Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing*, pages 289-299, December 1988.
- [6] P. Mehrotra and J. Van Rosendale. Programming distributed memory architectures using Kali. In A. Nicolau, D. Gelernter, T. Gross, and D. Padua, editors, *Advances in Languages and Compilers for Parallel Processing*, pages 364-384. Pitman/MIT-Press, 1991.
- [7] H. Zima, P. Brezany, B. Chapman, P. Mehrotra, and A. Schwald. Vienna Fortran - a language specification. Internal Report 21, ICASE, Hampton, VA, March 1992.

HIGH PERFORMANCE FORTRAN

1994 Workshop on The Role of Computers in LaRC R&D

June 16, 1994

Piyush Mehrotra

Institute for Computer Applications in Science and Engineering

pm@icase.edu

High Performance Fortran Forum

Primary goal:

to define a set of “standard” Fortran 90 extensions for *data parallel codes* to extract top performance on **SIMD** and **MIMD** non-uniform memory access (NUMA) machines.

549

Most scientific codes, including codes of interest to NASA researchers, exhibit data parallelism:

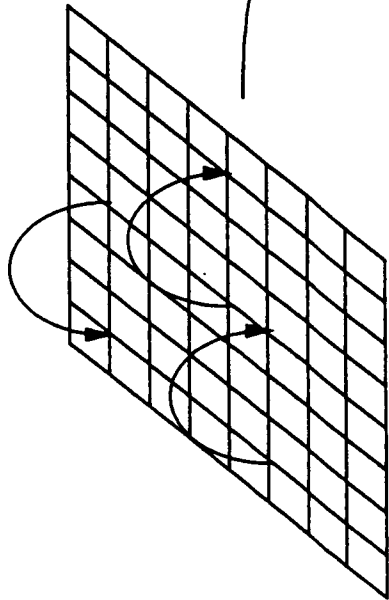
Data parallelism: *similar operations performed in parallel on different parts of structured data (represented by arrays).*

Motivation

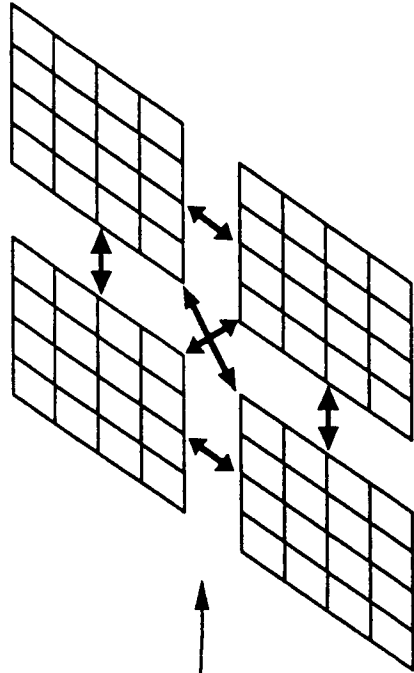
- Porting codes to fragmented memory machines requires:
 - distribution of data, and
 - insertion of communication.
- Explicitly parallel approach allows effective exploitation of underlying architecture.
- Disadvantages:
 - user has to cope with low level details
 - programs are difficult to design and debug
 - hardwires distribution choices

Approach:

User code - parallel operations
on shared data.



compiler



Allow programmers to use a global
name (index) space while controlling
the distribution of code and data.

Generated code - SPMD code with
embedded communication.

Approach

- Exploiting parallelism is a responsibility to be shared between the user and the compiler/runtime system.
- User controls data layout using a *global index space*.
- There are no explicit statements for process management, synchronization or communication.
- Compiler produces SPMD code with embedded statements for process interactions.

Previous work

IVTRAN - data distributions in an SIMD language for ILLIAC IV
(Mass. Comp. Assoc., 1973)

Kali - user specifiable data distributions for distributed memory machines (ICASE, 1988)

553 ... - *Dino, DPC, Superb, Crystal, ID Nouveau, CM Fortran, ...*

FortranD - allowed alignment to decompositions (Rice Univ., 1991)

Vienna Fortran - comprehensive set of extensions for Fortran (Univ. of Vienna/ICASE, 1992)

HPF Features

- Fortran 90 as base language - new data structures, array operations, modules, etc.
- User directives for data layout
- Data parallel constructs: forall statement and construct
- New intrinsics/library functions
- Extrinsic procedures
- *No* new I/O features
- Restrictions on sequence and storage association
- HPF subset

Data Layout Rationale

Memory of parallel architectures is generally fragmented. Thus:

- if unrelated data is properly *distributed*, i.e., in different memories, operations can be done in parallel:

```
DO i = 1, N
```

```
  A(i) = A(i) + 1
```

```
ENDDO
```

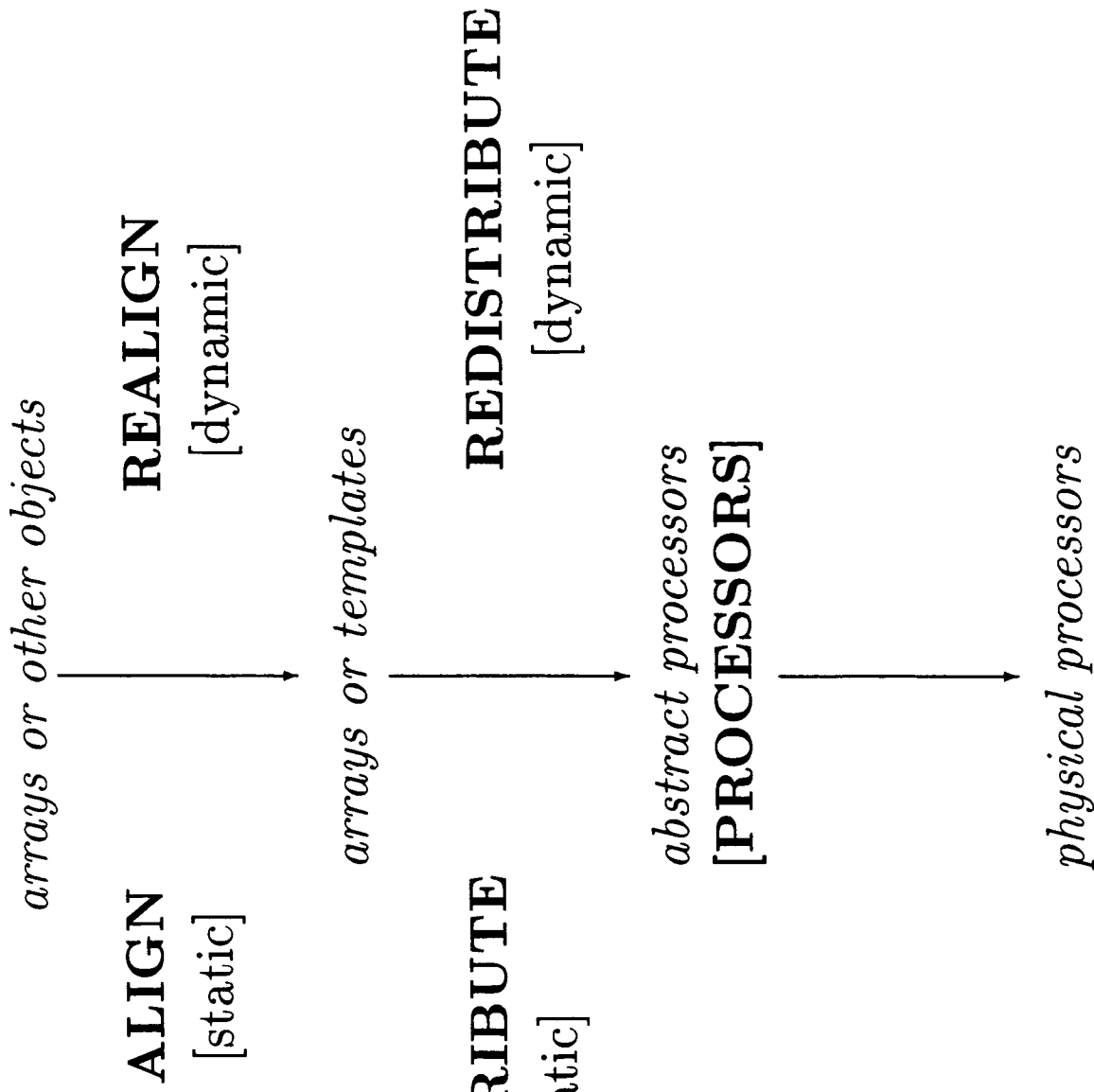
- if related data is properly *aligned*, i.e., in the same memory, communication costs can be reduced:

```
DO i = 1, N
```

```
  A(i) = B(i) + C(i+1)
```

```
ENDDO
```

HPF Mapping Model



Example - Jacobi

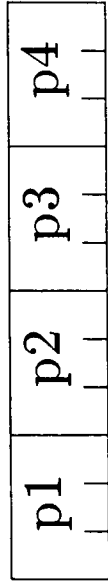
```
!HPF$ PROCESSORS p(number_of_processors())
REAL u(n, n), f(n, n)
!HPF$ ALIGN WITH u :: f
!HPF$ DISTRIBUTE (*, BLOCK) ONTO p :: u
...
FORALL (i = 2:n-1, j = 2:n-1)
    u(i, j) = 0.25*( u(i+1, j) + u(i-1, j)
                    + u(i, j+1) + u(i, j-1)) - f(i, j)
END FORALL
```

- p : a one-dimensional array of abstract processors
- array f identically aligned with array u
- columns of array u distributed blockwise
- code is written using a *global index space*
- compiler introduces required communication

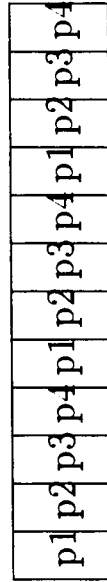
Distribution of Data

A 12 element array on 4 processors

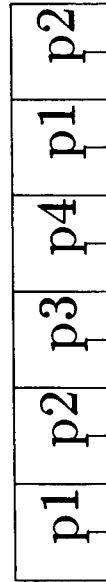
- **BLOCK :**



- **CYCLIC :**



- **CYCLIC (2):**



Example - Jacobi 2D distribution

```
!HPF$ PROCESSORS p(4,4)
REAL u(n, n), f(n, n)
!HPF$ ALIGN WITH u :: f
!HPF$ DISTRIBUTE (BLOCK, BLOCK) ONTO p :: u

FORALL (i = 2:n-1, j = 2:n-1)
    u(i, j) = 0.25*( u(i+1, j) + u(i-1, j)
+                   u(i, j+1) + u(i, j-1)) - f(i, j)
END FORALL
```

Only the distribution directives are changed - the code remains the same.

Conclusions

- A large percentage of scientific codes, including codes of interest to NASA researchers, are data parallel in nature.
- HPF provides a high level approach for porting data parallel algorithms to both shared and non-shared memory machines.
 - Add distribution directives to specify layout.
 - Use array statements and forall constructs to specify data parallelism.
- Porting of existing Fortran 77 codes may require extensive rewrites to eliminate dependence on storage and sequence association.
- Extrinsic procedures allow escape from the HPF model.

Further Information:

- Compilers with basic capabilities should be available soon.

Announced Products	Announced Efforts	Interested
Applied Parallel Research	TMC	Cray
Kuck and Associates	IBM	HP
Portland Group	nCube	Fujitsu
Intel	NEC	Silicon Graphics
Meiko	Maspar	Hitachi
Digital	Convex	Sun
	...	

- HPF document available by anonymous ftp from `titan.cs.rice.edu` in `/public/HPFF/draft`. Also as *Scientific Programming*, Vol. 2, Nos. 1-2, pages 1-170, Spring and Summer 1993.
- Mosaic URL: <http://www.erc.msstate.edu/hpff/home.html>
- Another round of HPFF meetings being held in Chicago currently for HPF 1 clarification and HPF 2 requirements.

SESSION 11 Advanced Topics

Chaired by

Susan Voigt

11.1 Current Research Activities at the NASA-sponsored Illinois Computing Laboratory of Aerospace Systems and Software - Kathryn Smith

11.2 Epistemology, SoftwareEngineering, and Formal Methods - C. Michael Holloway

**Current Research Activities at the NASA-sponsored
Illinois Computing Laboratory of
Aerospace Systems and Software**

P. 7

**Kathryn A. Smith
Assessment Technology Branch
Information and Electromagnetic Technology Division
Research and Technology Group**

The Illinois Computing Laboratory of Aerospace Systems and Software (ICLASS) is a NASA center for excellence in computer science. ICLASS was established in 1985 with two objectives:

- (1) to pursue research in the areas of aerospace computing systems, software and applications of critical importance to NASA; and
- (2) to develop and maintain close contacts between researchers at ICLASS and at various NASA centers to stimulate interaction and cooperation, and facilitate technology transfer.

Current ICLASS research activities are in the areas of parallel architectures and algorithms, reliable and fault-tolerant computing, real-time systems, distributed systems, software engineering, and artificial intelligence.



National Aeronautics and Space Administration
Langley Research Center

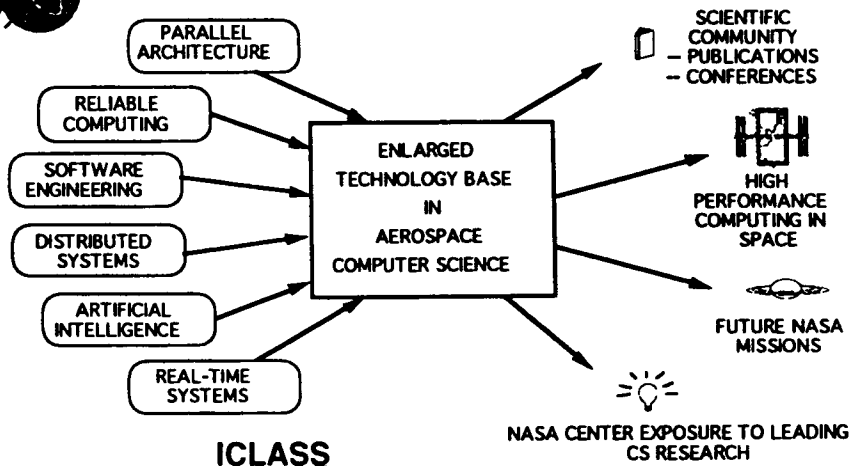
**Illinois Computing Laboratory for Aerospace
Systems and Software (ICLASS)**

- NASA Center for excellence, established 1985
- Objectives
 - Pursue research in aerospace computing systems, software and applications important to NASA
 - Develop close contacts and stimulate interactions between faculty and students at ICLASS and researchers at NASA
- Performance period Dec. 31, 1993- Dec. 30, 1994
- Annual review, May 24-25, 1993
 - Attended by 13 NASA people (2 centers & JPL), 1 US Navy, and 4 from industry
 - Panel presentation by NASA et al.
 - Presentations by ICLASS researchers
 - Poster sessions presented by students during breaks
- Over 70 recent publications

K. A. Smith



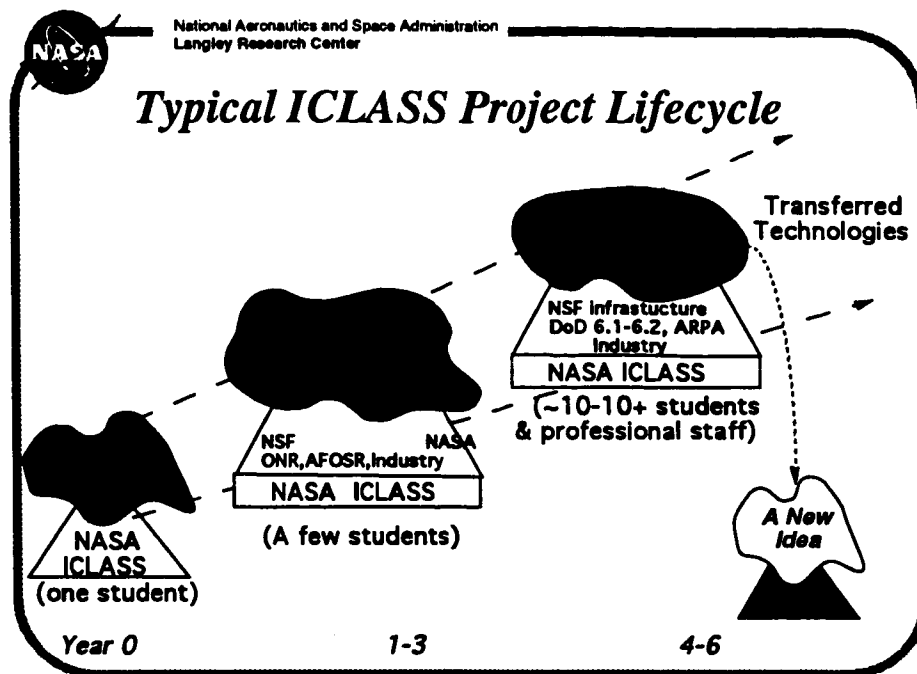
National Aeronautics and Space Administration
Langley Research Center



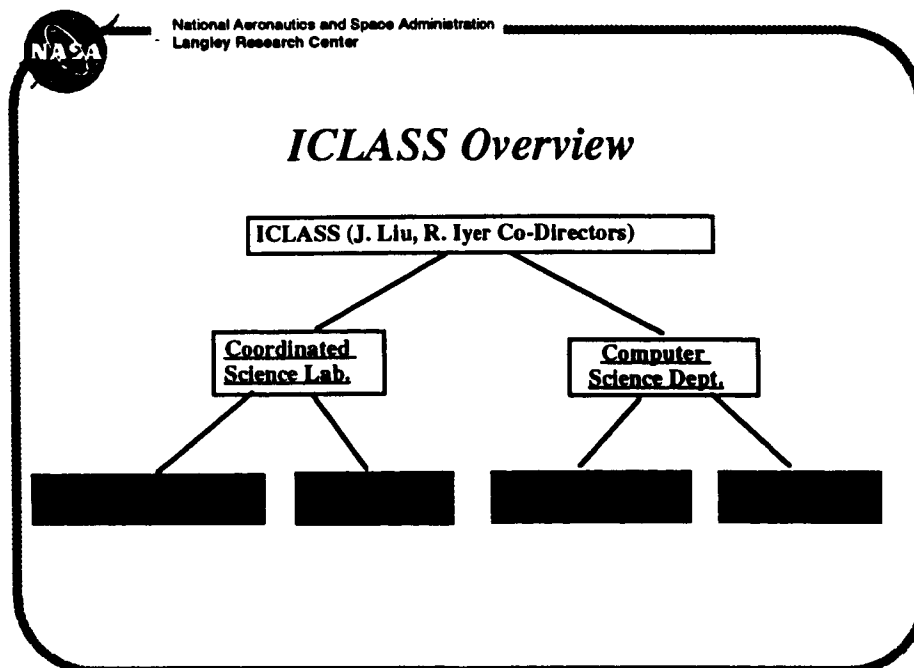
ICLASS

- Focus attention of CS researchers on NASA related problems
- Involve graduate students and research faculty
- Enhance NASA Computer Science understanding
- LaRC coordinates and maintains close technical communication

K. A. Smith



K. A. Smith



K. A. Smith



National Aeronautics and Space Administration
Langley Research Center

ICLASS RESEARCH ACTIVITIES - 1994



Development of Parallel Programs for Distributed Memory Multicomputers
Multiprocessor Architectures
Advanced Compilation and Architecture Technology
Resource Management for Parallel and Distributed Systems
Parallel System Performance Analysis
Efficient Execution of Fine-Grained Concurrent Programs
High Performance Memory Systems for Advanced Multiprocessors



Functional Programming and Scientific Computing
Formalization of Code Re-Use Through Abstract Algorithms
Three Dimensional Vision Systems
Engineering Integrated CAD/CAM Systems for Reduced Cost Manufacturing



Recovery in Dependable Parallel Architectures
A Design Environment for Fault Tolerant Systems
Dependability Validation of High Performance Systems
Verification of VHDL Digital Systems



Reliable, Distributed, Database Management Systems
Real-time Multiprocessor Operating Systems
A Prototype Environment for Real-Time Systems

K. A. Smith



National Aeronautics and Space Administration
Langley Research Center

Parallel Architecture and Algorithms

- **Development of Parallel Programs for Distributed Memory Multicomputers**
 - Design of efficient parallel algorithms to run on a variety of parallel architectures
 - Develop algorithms on top of abstract parallel programming framework (Chare Kernel)
- **Multiprocessor Architectures**
 - Develop, model and analyze high performance multiprocessor architectures that are fault tolerant and highly mission adaptive
 - Refine, test and port methodology for modeling and analyzing the performance of parallel processors under real workloads
- **Advanced Compilation and Architecture Technology**
 - Focus on the architecture and compiler techniques required to close the gap between peak performance and sustained performance of high performance multiprocessor systems
- **Resource Management for Parallel & Distributed Systems**
 - Develop more efficient algorithms for combinatorial searches on sequential and parallel computers

K. A. Smith



National Aeronautics and Space Administration
Langley Research Center

Parallel Architecture and Algorithms

- **Parallel System Performance Analysis**
 - Software tool set, Pablo, that supports source code performance instrumentation and graphical performance analysis
 - Focus of ICLASS student input-output performance optimization
- **Efficient Execution of Fine-Grained Concurrent Programs**
 - Focus on how to implement fine-grained, object-oriented concurrent programs to execute efficiently on a variety of parallel architectures, from small-scale, shared-memory multiprocessors to fine-grained, massively concurrent multicomputers
- **High-Performance Memory Systems for Advanced Multiprocessors**
 - Research aims at improving speed of large-scale multiprocessors
 - Focus of ICLASS student - memory hierarchy performance of the operating system

K. A. Smith



National Aeronautics and Space Administration
Langley Research Center

Reliable and Fault Tolerant Computing

- **Recovery in Dependable Parallel Architectures**
 - Develop new concepts in reliable memory management for dependable operation of parallel architectures
- **A Design Environment for Fault Tolerant Systems**
 - Investigate the development of a highly instrumented simulation-based CAD environment
 - Environment allows a designer to interactively evaluate the reliability and performance characteristics of proposed designs during the design process
- **Verification of VHDL Digital Systems**
 - Development of System-level verification tools capable of handling large systems
 - Emphasis on early detection of design errors
- **Dependability Validation of High Performance Systems**
 - Improve memory management performance in object-oriented, dynamically allocated, garbage collected virtual memory systems
 - Evaluate reliability by fault simulation

K. A. Smith



National Aeronautics and Space Administration
Langley Research Center

Software Engineering and Artificial Intelligence

- **Functional Programming and Scientific Computing**
 - Address the problems of expressiveness and efficiency in functional programming languages, emphasizing their use for scientific computation
- **Formalization of Code Re-Use Through Abstract Algorithms**
 - Study form and use of new abstraction method, using data structure independent algorithm skeletons
- **Three Dimensional Vision Systems**
 - Computer vision systems capable of three-dimensional interpretation of "flat" video images
- **Engineering Integrated CAD/CAM Systems for Reduced Cost Manufacturing**
 - Develop tools which improve interface between design and manufacturing
 - Move more manufacturability information into the design phase

K. A. Smith

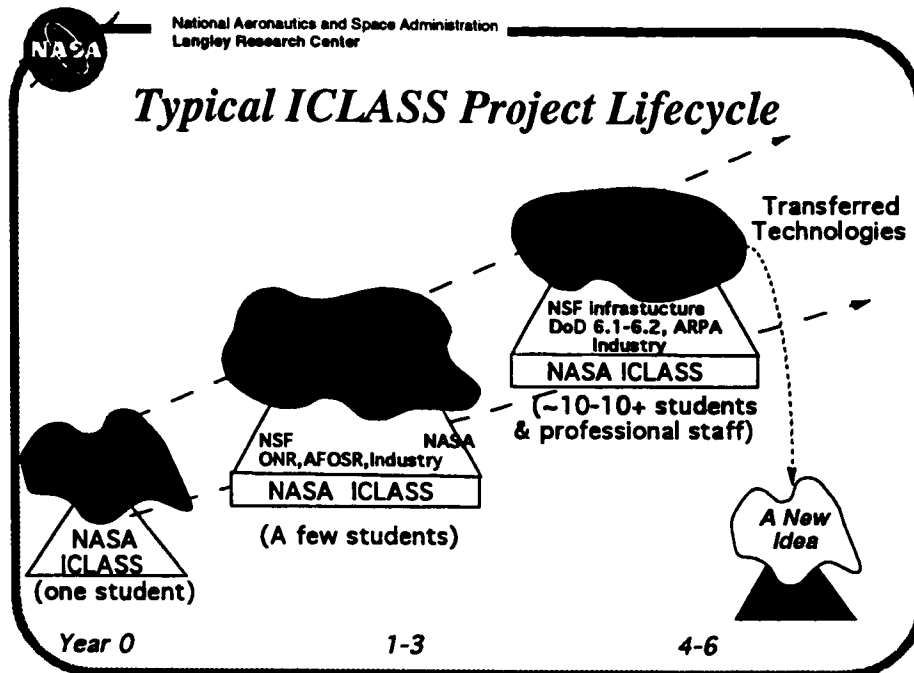


National Aeronautics and Space Administration
Langley Research Center

Distributed Systems and Real-Time Systems

- **Reliable, Distributed, Database Management Systems**
 - Design a reliable, distributed database management system
- **Multiprocessor Operating Systems**
 - Design and implement of customizable operating systems for real-time and high-performance multiprocessor applications
- **A Prototyping Environment for Real-Time Systems**
 - Build the Prototyping Environment for Real-Time Systems (PERTS)
 - PERTS an environment for
 - Evaluation of new design approaches
 - Experimentation with alternative system building blocks
 - Analysis and performance profiling of prototype real-time systems

K. A. Smith



K. A. Smith

NASA National Aeronautics and Space Administration
Langley Research Center

ICLASS Technology Transfer

DEPEND, Dependability Analysis Tool - IBM, Raytheon, Boeing

PERTS, Real-time Prototyping Tool - Tri-Pacific (to market), IBM, NASA Ames, NSWC

Pablo, Parallel System Performance - NASA GSFC, ARPA

Concert, Efficient Execution of Fine-Grained Concurrent Programs - Caterpillar

TEACHER, Resource Management for Parallel and Distributed Systems - NASA ARC

IMPACT, Compilation for Superscalar and Multiprocessor Architectures - HP Labs., Intel, Sun Labs.

For more information contact: Kathryn Smith, X41699
kas@sunspot.larc.nasa.gov

K. A. Smith

Epistemology, Software Engineering, and Formal Methods

Abstract of Presentation

C. Michael Holloway

P. 77

One of the most basic questions anyone can ask is, "How do I know that what I think I know is true?" The study of this question is called **epistemology**. Traditionally, epistemology has been considered to be of legitimate interest only to philosophers, theologians, and three-year-old-children, who respond to every statement by asking, "Why?" Software engineers need to be interested in the subject, however, because a lack of sufficient understanding of epistemology contributes to many of the current problems in the field.

Epistemology is a complex subject, one to which many philosophers and theologians have devoted their entire careers. The discussion here is necessarily brief and incomplete; however, it should be sufficient to demonstrate the critical importance of the subject to software engineering.

To the fundamental question of how do we know what is true, there are three basic answers: authority, reason, and experience. An epistemology based on *authority* states that truth is given to us by someone more knowledgeable than ourselves. The two primary variations of authority-based epistemologies are *omniscient authority* (the authority is God), and *human authority* (the authority is a human expert).

An epistemology based on *reason* claims that what is true is that which can be proven using the rules of deductive logic. Finally, an epistemology based on *experience* claims that what is true is that which can be encountered through one or more of the senses.

Several different variations of experience-based epistemologies exist. The two variations relevant to this discussion are *anecdotal experience* and *empirical evidence*. The first states that truth for any particular individual or group of individuals is that which the individual, or group, personally experiences. The second states that truth is that which can be verified through carefully controlled experiments.

The relative strengths of these epistemological approaches are as follows. Omniscient authority provides absolute truth; if there is a God and He has spoken on something, then what He says must, by definition, be true. Reason yields conditional absolute truth; if the premises on which a valid deductive argument are known to be true, then the conclusion of the argument must also be true.

Empirical evidence provides probable truth; if controlled experiments are designed properly and repeated enough times, then it is highly probable that the results

accurately describe reality. Anecdotal experience yields possible truth; if something happened for one person, it is possible it might happen to others also. Finally, human authority provides opinion.

On which of these approaches to epistemology is software engineering mostly based?

The software engineering literature is filled with pronouncements about how software should be developed (e.g., "*Object-oriented development is the best way to obtain reusable software*"). Rarely, if ever, are these pronouncements augmented with either logical or experimental evidence. Thus, one is forced to conclude that much of software engineering is based on a combination of anecdotal experience and human authority. That is, we know that a particular technique is good because John Doe, who is an expert in the field, says that it is good (human authority); John Doe knows that it is good because it worked for him (anecdotal experience). This is a weak epistemological foundation on which to base an entire discipline.

This current state should not be surprising; the development of software engineering is following the same pattern as the development of many other disciplines. Civil engineering, chemical engineering, aeronautical engineering, and others all had periods in which they relied almost exclusively on anecdotal experience and the subsequent authority of the "experts". Often, it took major disasters before practitioners in such fields began to investigate fully the foundations on which their field was based.

To date, although there have been many, many software problems, there have been no major disasters that have been directly attributed to software. However, unless a sound epistemological foundation is established for software engineering, disasters will come one day. To avoid this, research is needed to develop valid approaches to answering questions about both software products (e.g., are these requirements consistent?) and software processes (e.g., is method A better than method B?).

The Assessment Technology Branch (ATB), which is part of the Information and Electromagnetic Technology Division, Research and Technology Group, is currently investigating empirical methods to answer process-type questions and logical methods to answer product-type questions. The remainder of the presentation discusses the second of these two avenues of research.

A team led by Ricky W. Butler has been studying the discipline of *formal methods* for over 6 years. Other civil-servants on the team are Jim L. Caldwell, Victor A. Carreño, C. Michael Holloway, and Paul S. Miner. Vigyan, Inc., Stanford Research Institute International (SRI), Odyssey Research Associates (ORA), and Computational Logic, Incorporated (CLI) conduct research under contract.

Formal methods is¹ the applied mathematics of computer systems engineering². Formal methods aims to be to software engineering what fluid dynamics is to aeronautical engineering and what classical mechanics is to civil engineering. The mathematics of formal methods includes predicate calculus (first order logic), recursive function theory, lambda calculus, programming language semantics, and discrete mathematics (e.g., number theory, abstract algebra). To this mathematical base, formal methods adds notions from programming languages such as data types, module structure, and generics.

There are many different types of formal methods with various degrees of rigor. The following is a useful classification of the possible degrees of rigor in the application of formal methods:

- Level 0: No use of formal methods
- Level 1: Formal specification (using mathematical logic or a specification language with formal semantics) of all or part of a system
- Level 2: Formal specification at two or more levels of abstraction and paper-and-pencil proofs that the detailed specification satisfies the abstract one
- Level 3: Like level 2, except paper-and-pencil proofs are replaced by formal proofs checked by a semi-automatic theorem prover.

Presently, a complete (level 3) verification of a large, complex system is impractical; however, application of formal methods to critical portions of a system is practical and useful.

The specification of a simple phone book provides a suitable simple example of many of the basic ideas and benefits of formal methods. Please see the presentation visuals that follow this abstract for this example.

Because of the promise that formal methods offers, a considerable amount of high-quality research is being conducted or sponsored by ATB. This research includes, but is not limited to, the following projects:

- Detailed design with complete level 3 verification of the Reliable Computing Platform, which is a fault-tolerant computing base able to recover from both permanent and transient faults
- Design with level 2/3 verification of a transient fault-tolerant clock synchronization circuit; this circuit has also been fabricated, but the layout was done by hand without formal verification
- In cooperation with SRI and Rockwell-Collins, level 3 specification and verification of the microcode of the AAMP5 microprocessor
- In cooperation with ORA and Union Switch and Signal, level 3 specification and verification of a next-generation railroad control system
- Under contract, ORA is working with Honeywell on level 3 specification and verification of aircraft navigation functions
- Under contract, Vigyan and SRI are working with Loral, Johnson Space Center, and the Jet Propulsion Laboratory on level 3 specification and verification of some Space Shuttle functions
- Under contract, SRI is working with Allied-Signal on level 3 specification and verification of important algorithms for fault-tolerance

In addition to these, and other, projects, the branch conducts periodic workshops on formal methods. Previous ones were held in 1990 and 1992; the next one is planned for 1995. Also, an extensive collection of information on the research is available through the World Wide Web at the following Universal Resource Locator:

<http://shemesh.larc.nasa.gov/fm-top.html>

Interested individuals are encouraged to explore this collection.

A lot of ground has been covered in this presentation, but the most important point is simple:

Epistemology: It's important, learn about it
Software Engineering: It's immature, work on it
Formal Methods: It's promising, look for it

1. Just like mathematics, formal methods should be treated as a singular, not plural, noun.

2. The ideas apply equally well to both software and complex hardware devices.

Epistemology Software Engineering and Formal Methods

C. Michael Holloway

*Assessment Technology Branch
Information & Electromagnetic Technology Division
Research & Technology Group
Langley Research Center
National Aeronautics and Space Administration
United States Government*

The Role of Computers in LaRC R & D (June 16-18, 1994)

Introduction

- One of the most basic questions anyone can ask is
“How do I know that what I think I know is true?”
- The study of this question is called ***epistemology***
- Traditionally, epistemology has been considered to be of legitimate interest only to philosophers, theologians, and three-year-old children
- At least one other group should be very interested in epistemology – software engineers – because lack of understanding in this area plagues the field

The Basics of Epistemology

- . There are three basic answers to the question of how do we know what is true
 - *Authority*: truth is given to us by a knowledgeable person
 - *Reason*: truth is what can be proven using the rules of deductive logic
 - *Experience*: truth is what can be encountered through one or more of the senses
 - * *Anecdotal experience*: truth is what an individual or a group of individuals experiences personally
 - * *Empirical evidence*: truth is what can be verified through carefully controlled experiments

Examples of Truth by Authority

- . The Ten Commandments
(*omniscient authority*)
- . 1-year-old, pointing to the family cat:
“Whatsthat?”
father: “Kitty”
(*human authority*)

Examples of Truth by Reason

- . If that creature is a tove, then it is slithy
That creature is a tove
Therefore, that creature is slithy
- . If the airplane was built by Boeing, then it is a jet
The airplane is not a jet
Therefore, the airplane was not built by Boeing
- . $X + Y = 7$
 $3Y - 2X = 1$
Therefore, $X = 4$ and $Y = 3$

Examples of Truth by Anecdotal Experience

- . Smoking doesn't shorten your life because my father smoked all his life and lived to be 95.
- . Whenever I have the hiccups, I hold my breath and count to 10 and they go away. Therefore, holding your breath and counting to 10 cures the hiccups.
- . We used method M and had 40% fewer bugs in testing. You should use method M, too.

Examples of Truth by Empirical Evidence

- . The dive-recovery flap for the P38 in World War II developed through tests in Langley's 8-Foot High Speed Tunnel
- . 5,000 patients were given drug X. 5,000 patients were given no drugs at all. 4,998 of the patients given drug X got better within 1 week. 3 of the patients given no drugs at all got better within 1 week. Drug X helps.

Relative Strengths

- . Omniscient Authority: *absolute truth*
- . Reason: *conditional absolute truth*
- . Empirical Evidence: *probable truth*
- . Anecdotal Experience: *possible truth*
- . Human Authority: *opinion*

How Does This Apply to Software Engineering?

- The software engineering community is full of claims
 - "The best way to develop reusable software is to use object-oriented design."*
 - "Programmers should never be allowed to test their own code."*
 - "Getting control of the software process is the key – SEI's CMM is the way to do this."*
 - "We need more standards!"*
 - "Much progress has been made in the last few years in improving the way we develop software."*
 - "GOTO's are harmful."*
 - "CASE tools are the best way to improve software productivity."*
- Many people accept these, or other similar, claims as being true

The Fundamental Question

What is the
epistemological foundation
for accepting
these claims?

The Answer

- Logically sound arguments are rarely given
- Virtually no empirical evidence is cited
- Instead, software engineering is based almost entirely on a combination of human authority and anecdotal experience
 - *We know that technique C is good because Jane Doe, who is a recognized authority in the field, says that it is good (human authority)*
 - *Jane Doe knows that it is good because she used it on a project once and got good results (anecdotal experience)*
- This is a weak epistemological foundation, one on which no legitimate claims of success can be based

Implications of This Epistemological Weakness

- Until we get adequate evidence, we should be very cautious in the claims we make and the standards we set
 - It is fine to say, *"Method M seems to have improved our productivity, so you might want to try it."*
But it is dishonest to say, "If you want to improve your productivity, you must use Method M."
 - *"Company R used method F and found errors they don't think they would have found using their old methods,"* is fine
"Method F finds errors that other methods do not find," is dishonest
- The software engineering community should be investigating methods for obtaining strong (that is, logical or empirical) evidence

Why Has More Not Been Done?

- The development of software engineering is following the same pattern as the development of other disciplines
 - *Civil engineering, chemical engineering, aeronautical engineering, etc. all had periods in which they relied almost exclusively on anecdotal evidence*
 - *Often, it took major disasters to prompt changes*
- It is hard
- It is expensive
- It is not glamorous
- Few people care: We haven't had a *major* disaster yet

Why Must More Be Done?

- Without adequate evidence, we are easily influenced by the latest bandwagon that goes rumbling by
- Without adequate evidence, we may well "cast-in-concrete" something that ought not even be "cast-in-mud"
- Without adequate evidence, the following two statements are equally as meaningless:
 - *You shall use method M in developing your software*
 - *'Twas brillig by the slithy tove*
- Without adequate evidence, disasters are inevitable

Towards Establishing a Valid Epistemological Foundation

- Recognize the fundamental need for such a foundation
- Understand the different approaches needed for *process* and *product*
 - Process questions (e.g., *Is method A better than method B?*) need to be answered empirically
 - Product questions (e.g., *Are my requirements consistent?*) need to be answered by an appropriate combination of logical and empirical methods
- Refuse to accept claims based on insufficient evidence

Current Research at LaRC

- Kelly Hayhurst (IETD/ATB) is leading an effort to develop an empirical evaluation of a particular approach to IV & V
For more information, contact Kelly
Email: k.j.hayhurst@LaRC.NASA.GOV
Phone: 46215
- The formal methods team led by Ricky Butler (IETD/ATB) is investigating logical methods for answering product-type questions
 - *Other team members are Jim Caldwell, Victor Carreño, Michael Holloway, and Paul Miner*
 - *Remainder of talk concerns this work*

Further Reading on Epistemology

- If you are interested in more information on epistemology, I recommend you start with the following two books:
 - *Thales to Dewey*, by Gordon H. Clark, 2nd edition, 1989, ISBN 0-940931-26-5
 - *The Philosophy of Science*, by Gordon H. Clark, 2nd edition, 1987, ISBN 0-940931-18-4
- These two books contain pointers to most of the important philosophical works throughout the ages

Singular or Plural?

- Which of the following is correct?

Formal methods is the applied mathematics ...

OR

Formal methods are the applied mathematics ...

- Answer depends on the writer or speaker
- I will tend to use “formal methods” as singular

What is Formal Methods?

- Formal methods is the applied mathematics of computer systems engineering
- The mathematics of formal methods includes:
 - *predicate calculus (1st order logic)*
 - *recursive function theory*
 - *lambda calculus*
 - *programming language semantics*
 - *discrete mathematics: number theory, abstract algebra, etc.*

What is Formal Methods? (continued)

<i>System Designed</i>	<i>Engineering</i>	<i>Theory</i>
Bridge	Civil	Classical Mechanics
Airframe	Aeronautical	Fluid Dynamics
Nuclear Reactor	Nuclear	Quantum Mechanics
Digital Avionics System	Software	Formal Methods

Classical vs Computer Systems

<i>Classical Systems</i>	<i>Computer Systems</i>
continuous state space	discrete state space
smooth transitions	abrupt transitions
finite testing & interpolation acceptable	finite testing inadequate, interpolation unsound
mathematical modeling	prototyping & testing
build to withstand additional stress	build to specific assumptions
predictable	surprising

What Makes a Technique a Formal Method?

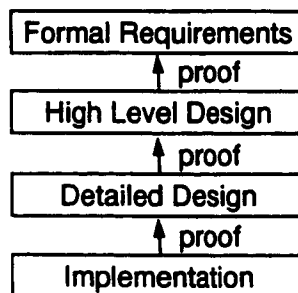
- Formal method = logic + programming language concepts
- Important attributes:
 - *logic based*
 - *programming language concepts (e.g., data types, module structure, generics)*
 - *fully and formally specified semantics*
 - *should be able to express what is done without saying how it is done (i.e., non-procedural)*
 - *supports the building of useful tools for analysis*

Levels of Rigor of Formal Methods

- *Level 0*: No use of formal methods
- *Level 1*: Formal specification (using mathematical logic or a specification language with formal semantics) of all or part of a system
- *Level 2*: Formal specification at two or more levels of abstraction and paper-and-pencil proofs that the detailed specification satisfies the abstract one
- *Level 3*: Like level 2, except paper-and-pencil proofs are replaced by formal proofs checked by a semi-automatic theorem prover

Extent of Application

- Formal Methods is not an all-or-nothing approach
- Complete formal verification of a large complex system is impractical at this time



- Application of formal methods to critical portions of a system is practical and useful

Extent of Application (example)

- In the Reliable Computing Platform, we use formal methods to establish:

ENOUGH_WORKING_HARDWARE

⊃

PROPER_OPERATION

- We use reliability analysis to calculate:

Probability[ENOUGH_WORKING_HARDWARE]

- Reliability analysis relies on physical testing of devices to establish some important parameters

Level 1 Example: Phone Book English Requirements

- The phone book shall store the phone numbers of a city
- Given a name, there shall be a way to retrieve an associated phone number
- It shall be possible to add and delete entries from the phone book

Level 1 Example: Phone Book

Choosing a Specification Approach

- How do we represent the phone book mathematically?
 1. A set of ordered pairs $\langle \text{name}, \text{number} \rangle$. Adding and deleting entries is by set addition and deletion.
 2. A function whose domain is all possible names and range is all phone numbers. Adding and deleting entries is by modification of function values.
 3. A function whose domain is only names currently in the phone book and range is phone numbers. Adding and deleting entries is by modification of the function domain and values. (Z style)
- We choose to use approach 2

Level 1 Example: Phone Book

Specifying the Book

- Using traditional mathematical notation, we would write:

Let N = set of names

P = set of phone numbers

$book: N \rightarrow P$

- To indicate that we do not have a phone number for all possible names, but only for names of real people, we decide to use a special number: $\rho \in P$
- An empty phone book is specified as follows:

$emptybook: N \rightarrow P$

$emptybook(nm) \equiv \rho$

Level 1 Example: Phone Book Accessing an Entry

Let N = set of names

P = set of phone numbers

$book: N \rightarrow P$

B = set of functions: $N \rightarrow P$

$FindPhone: B \times N \rightarrow P$

$FindPhone(bk, name) = bk(name)$

Note that $FindPhone$ is a higher-order function, because its first argument is a function

Level 1 Example: Phone Book Adding/Deleting an Entry

Let N = set of names

P = set of phone numbers

$book: N \rightarrow P$

$\rho \in P$

B = set of functions: $N \rightarrow P$

$AddPhone: B \times N \times P \rightarrow B$

$$AddPhone(bk, name, num)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ num & \text{if } x = name \end{cases}$$

$DelPhone: B \times N \rightarrow B$

$$DelPhone(bk, name)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ \rho & \text{if } x = name \end{cases}$$

Level 1 Example: Phone Book Complete Specification

Let N = set of names
 P = set of phone numbers
 $book: N \rightarrow P$
 $\rho \in P$
 B = set of functions: $N \rightarrow P$

$emptybook: N \rightarrow P$
 $emptybook(nm) \equiv \rho$

$FindPhone: B \times N \rightarrow P$
 $FindPhone(bk, name) = bk(name)$

$AddPhone: B \times N \times P \rightarrow B$
 $AddPhone(bk, name, num)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ num & \text{if } x = name \end{cases}$

$DelPhone: B \times N \rightarrow B$
 $DelPhone(bk, name)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ \rho & \text{if } x = name \end{cases}$

Level 2 Example: Phone Book Putative Theorems

A putative theorem is a theorem that we know must be true if we have formulated the specification correctly.

Lemma putative 1:

$FindPhone(AddPhone(bk, name, num), name) = num$

Proof:

$FindPhone(AddPhone(bk, name, num), name) = num$

$AddPhone(bk, name, num)(name) = num$

$num = num$

Q.E.D.

Level 2 Example: Phone Book Putative Theorems (continued)

Lemma putative 2: $bk(name) = \rho \supset$
 $DelPhone(AddPhone(bk, name, num), name) = bk$

Lemma putative 3: $(\forall i: name_i \neq name) \wedge$
 $book = AddPhone(bk, name, num) \wedge$
 $book_1 = AddPhone(book, name_1, num_1) \wedge$
 $book_2 = AddPhone(book_1, name_2, num_2) \wedge$
 \vdots
 $book_n = AddPhone(book_{n-1}, name_n, num_n)$
 \supset
 $FindPhone(book_n, name) = num$

Formal methods can establish that a property holds even in the presence of an arbitrary number of operations; testing can never establish this.

Level 3 Example: Phone Book PVS Specification

```

phonebook: THEORY
BEGIN

  names: TYPE
  name0: names
  ph_number: TYPE
  p_0: ph_number
  book: TYPE = [names -> ph_number]

  name: VAR names
  emptybook(name): ph_number = p_0
  bk: VAR book

  FindPhone(bk, name): ph_number = bk(name)

  num: VAR ph_number
  AddPhone(bk, name, num): book = bk WITH [name := num]

  DelPhone(bk, name): book = bk WITH [name := p_0]

  putative_1: LEMMA FindPhone(AddPhone(bk, name, num), name) = num
  putative_2: LEMMA bk(name) = p_0 IMPLIES
    DelPhone(AddPhone(bk, name, num), name) = bk

END phonebook

```

Level 3 Example: Phone Book Proof Using PVS

putative_1 :

```
|-----  
{1} (FORALL (bk: book), (nm: names), (num: ph_number):  
      FindPhone(AddPhone(bk, nm, num), nm) = num)
```

Rule? (skosimp*)
Repeatedly Skolemizing and flattening,
this simplifies to:
putative_1 :

```
|-----  
{1} FindPhone(AddPhone(bk!1, nm!1, num!1), nm!1) = num!1
```

Rule? (expand "FindPhone")

Level 3 Example: Phone Book Proof Using PVS (continued)

Rule? (expand "FindPhone")
Expanding the definition of FindPhone, this simplifies to:
putative_1 :

```
|-----  
{1} AddPhone(bk!1, nm!1, num!1)(nm!1) = num!1
```

Rule? (expand "AddPhone")
Expanding the definition of AddPhone, this simplifies to:
putative_1 :

```
|-----  
{1} TRUE
```

which is trivially true.
Q.E.D.

Run time = 1.02 secs.
Real time = 20.00 secs.

Level 1 Example: Phone Book Deficiencies in the Specification

- Our specification does not rule out the possibility of someone having a “p” phone number
- We have not allowed multiple phone numbers per single name
- Our specification does not say anything about whether there should be a warning if a deletion is requested on name that is not in the phone book

How do we remedy these deficiencies?

Level 1 Example: Phone Book Overcoming Deficiencies 1 & 2

Let N = set of names
 P = set of phone numbers
 $book: N \rightarrow 2^P$
 B = set of functions: $N \rightarrow 2^P$

$emptybook(name) \equiv \emptyset$

$FindPhone: B \times N \rightarrow 2^P$
 $FindPhone(bk, name) = bk(name)$

$AddPhone: B \times N \times P \rightarrow B$
 $AddPhone(bk, name, num)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ bk(name) \cup \{num\} & \text{if } x = name \end{cases}$

$DelPhone: B \times N \rightarrow B$
 $DelPhone(bk, name)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ \emptyset & \text{if } x = name \end{cases}$

Level 1 Example: Phone Book

An Additional Deficiency

- Notice that the function *DelPhone* deletes all of the phone numbers associated with a name
- There is no way to remove just one of the phone numbers that is associated with a given name
- The original requirements did not address this situation; to address it, we must define an additional function:

DelPhoneNum: $B \times N \times P \rightarrow B$

$$\text{DelPhoneNum}(bk, name, num)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ bk(name) \setminus \{num\} & \text{if } x = name \end{cases}$$

Example: Phone Book

Revised Requirements

Original Requirements

- The phone book shall store the phone numbers of a city
- Given a name, there shall be a way to retrieve an associated phone number
- It shall be possible to add and delete entries from the phone book

Revised Requirements

- For each name in the city, a set of phone numbers shall be stored
- Given a name, there shall be a way to retrieve the associated phone numbers
- It shall be possible to add a new name and phone number
- It shall be possible to add new phone numbers to an existing name
- It shall be possible to delete a name from the phone book
- It shall be possible to delete one of the phone numbers associated with a name
- A warning need not be given for a requested deletion of a name not in the city
- A warning need not be given for a requested deletion of a non-existent phone number

Example: Phone Book Observations

- Our specification is abstract. The functions are defined over infinite domains.
- In translating the requirements from English into a more formal notation, many things that were left out of the English were explicitly enumerated.
- The formal process exposed ambiguities and deficiencies in the requirements. E.g., we had to choose between

$$\textit{book}: N \rightarrow P$$

$$\textit{book}: N \rightarrow 2^P$$

- Putative theorem proving and scrutiny revealed deficiencies in the formal specification

Example: Phone Book More Observations

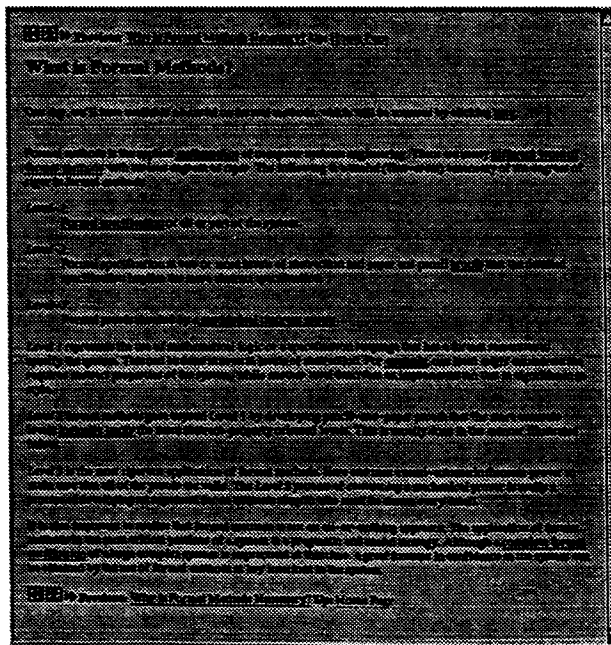
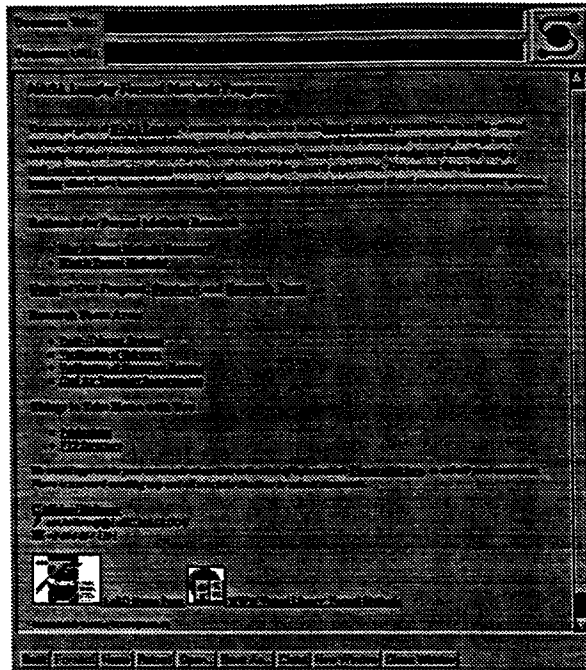
- There are many different ways to formally specify
- No matter what representation you chose you are making some decisions that bias the implementation
- The goal is to minimize this bias and yet be complete
- The process of formalizing the requirements can reveal problems and deficiencies and lead to a better English requirements document also
- The formal specification process is similar to the mathematical modeling process of engineering disciplines

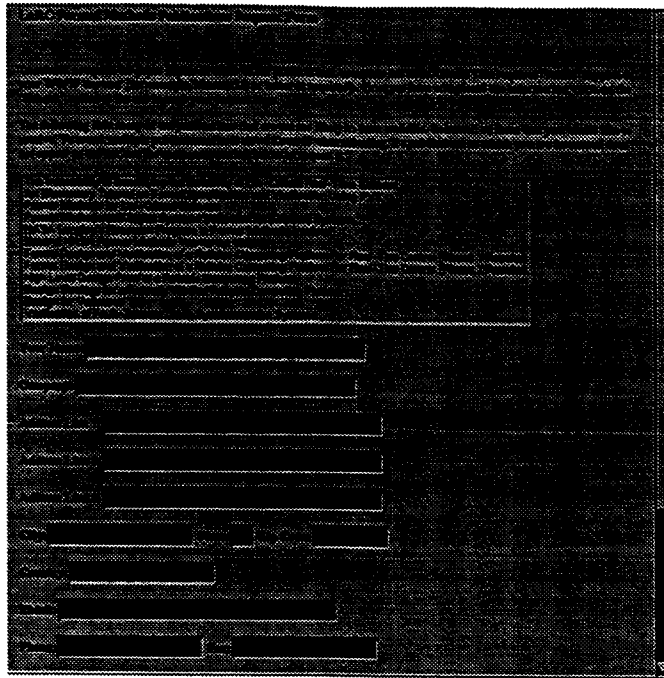
Formal Methods Research at LaRC

- Detailed design with complete level 3 verification of a Reliable Computing Platform
- Design with level 2/3 verification of a transient fault-tolerant clock synchronization circuit and fabrication of the circuit
- With SRI International & Rockwell-Collins, level 3 specification and verification of the microcode of the AAMP5 microprocessor
- With Odyssey Research Associates & Union Switch and Signal, level 3 specification and verification of next-generation railroad control system
- ORA & Honeywell, level 3 specification and verification of aircraft navigation functions

Formal Methods Research at LaRC (continued)

- Vigyan & SRI working with Loral, JSC, JPL on level 3 specification and verification of some Space Shuttle functions
- SRI working with Allied-Signal on level 3 specification and verification of important algorithms for fault-tolerance
- Conduct periodic workshops on formal methods; previous ones in 1990, 1992, with next one planned for 1995
- Maintain extensive collection of information on the research, accessible through the World Wide Web at URL
<http://shemesh.larc.nasa.gov/fm-top.html>





Epistemology
It's Important, Learn About It

Software Engineering
It's Immature, Work On It

Formal Methods
It's Promising, Look For It

REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE October 1994	3. REPORT TYPE AND DATES COVERED Conference Publication	
4. TITLE AND SUBTITLE The Role of Computers in Research and Development at Langley Research Center			5. FUNDING NUMBERS WU 505-90-53	
6. AUTHOR(S) Carol D. Wieseman, Compiler				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-0001			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001			10. SPONSORING / MONITORING AGENCY REPORT NUMBER NASA CP-10159	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 62			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>This document is a compilation of the presentations given at the workshop, "The Role of Computers in Research and Development at Langley Research Center," on June 15-16, 1994. The objectives of the workshop sponsored by the Computer Systems Technical Committee were to inform the LaRC community of the current software system and software practices being used at LaRC. To meet these objectives, there were talks presented by members of the Langley community, Naval Surface Warfare Center, Old Dominion University, and Hampton University.</p> <p>The workshop was organized in 10 sessions as follows: Software Engineering; Software Engineering Standards, Methods, and CASE Tools; Solutions of Equations; Automatic Differentiation; Mosaic and the World Wide Web; Graphics and Image Processing; System Design Integration; CAE Tools; Languages; and Advanced Topics.</p>				
14. SUBJECT TERMS Software Engineering			15. NUMBER OF PAGES 604	
			16. PRICE CODE A99	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	