2

ISI/SR-85-150

University of Southern California

1984 ANNUAL TECHNICAL REPORT

July 1983 - June 1984

A Research Program in Computer Technology

INFORMATION SCIENCES INSTITUTE

213/822-1511

4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ISI/SR-85-150	2. GOVT ACCESSION NO. AD-A157 991	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* 1984 Annual Technical Report: A Research Program in Computer Technology		5. TYPE OF REPORT & PERIOD COVERED Annual Technical Report July 1983 – June 1984 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) ISI Research Staff		8. CONTRACT OR GRANT NUMBER(s) F49620-84-C-0100 MDA903 81 C 0335 MCS-8304190
9. PERFORMING ORGANIZATION NAME AND ADDRESS USC/Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE April 1985 13. NUMBER OF PAGES 159
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

This document is approved for public release and sale; distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

....

18. SUPPLEMENTARY NOTES

....

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

1. domain model, domain principles, expert systems, integration knowledge, LISP, NIKL paraphraser, optimization knowledge, program writer, XPLAIN system
2. artificial intelligence, computing environments, constraints, knowledge representation, programming environments, rapid prototyping, software modeling, specification languages

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This ~~report~~ summarizes the research performed by USC/Information Sciences Institute from July 1, 1983, to June 30, 1984, for the Defense Advanced Research Projects Agency, the Air Force Office of Scientific Research, and the National Science Foundation. The research is focused on the development of computer science and technology, which is expected to have a high DoD/military impact.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

11. CONTROLLING OFFICE NAME AND ADDRESS (continued)

National Science Foundation
1800 G Street NW
Washington, DC 20550

Air Force Office of Scientific Research
Building 410, Bolling Air Force Base
Washington, DC 20332

19. KEY WORDS (continued)

3. automation-based paradigm, GIST specification language, granularity, high-level editing, locally formal notations, mappings, reimplementations, software development, software maintenance, software specification, Will programming language
4. formal development structure, grammar-based editing, GIST, Paddle, Popart, reimplementations, replay, structure comparison
5. design rules, device fabrication service, device testing, integrated circuit oriented language, MOSIS–MOS Implementation System, silicon compilation, standard pad frames, VLSI design, VLSI design library, wafer testing
6. CMOS/Bulk fabrication, MOSIS, printed circuit board fabrication, scalable design rules, VLSI
7. briefing aid application program, command graphics, computer graphics, high-level graphics language, network-based graphics, online map display
8. computer mail, gateways, interconnection, internetwork protocol, multimedia mail, networks, protocol design, protocols, protocol verification, simple mail transfer protocol, transmission control protocol, type-of-service
9. expert systems, intelligent systems, man-machine interfaces, pilot's associate, Strategic Computing Initiative
10. automated user interface, information dissemination, online document distribution, Strategic Computing research community
11. computer network, digital voice communication, network conferencing, packet satellite network, packet-switched networks, packet video, packet voice, secure voice transmission, signal processing, speech processing, vocoding
12. implementation of interactive systems, knowledge base, knowledge-based inference, natural interface, online services, process script, service building, tool building, user interface
13. bit-mapped graphics, consistent underlying environment, COUSIN, extensible environment, integration, interactive devices, knowledge-based approach, MENUNIX, menus, natural language, semantic model, service building, service execution, windows
14. artificial intelligence, discourse modeling, functional linguistics, grammar development, human-computer interfaces, natural language, Nigel, Penman, Systemic Linguistics, text generation, text planner, text structure
15. application software, ARPANET, computer network, hardware, Interlisp, KA/LI, KL10/KL20, network services, operations, PDP11/45, QLISP, resource allocation, system software, TOPS-20, UNIX, upgrades, VAX 11/750-80, VMS
16. distributed processing, gateways, large disk servers, local area networks, long-haul network connections, portable workstations
17. distributed processing, portable workstations, survivable networks
18. internetwork protocol, transmission control protocol, protocol implementation
19. address space limitations, C, Interlisp, LISP dialects, UNIX, VAX, VMS
20. distributed processing, local networks, personal computers, workstation environment
21. ARPANET/MILNET access, host systems, network services support
22. computer communication networks, packet radio, survivable networks, system support

20. ABSTRACT (continued)

The ISI program consists of 22 research areas:

Explainable Expert Systems: creating a framework for building expert systems that enhances an expert system's explanatory capablities (allowing it to explain and justify its reasoning), and eases modification and maintenance of an expert system: *Information Management*: combining artificial intelligence, software technology, and database techniques to build a more highly integrated, uniform, and user-evolvable computing environment for both software development and information services; *Mappings*: development of transformations for converting high-level specifications of programs into implementations in software or VLSI; *Transformation-Based Maintenance*: supporting reimplementations of a framework that allows a programmer to formally develop software from specifications via transformations by automatically supplying sequences of low-level transformations which accomplish or facilitate high-level transformations: *VLSI*: providing a low-cost, fast turnaround LSI/VLSI device fabrication service to support a geographically distributed VLSI research community with no direct access to VLSI fabrication but with access to a computer communication network, and conducting research on the VLSI design problem; *Advanced VLSI*: providing access to 1.2 micron CMOS/Bulk fabrication, printed circuit board fabrication, and centralized functional testing of custom-designed parts prior to their assembly into packages and boards; *Command Graphics*: development of a device-independent graphics system and graphics-oriented command and control applications programs; *Internet Concepts Research*: exploring aspects of protocols for the interconnections of computer communication networks, specifically the design and implementation of an internetwork computer message system and the design of internetwork host and gateway protocols: *NeaT*: development of an advanced, information-oriented man-machine interface for aircraft, which will behave as an intelligent associate to the pilot—the interface will be based on an expert system, which models the decisions of an experienced copilot; *Strategic Computing Information System*: providing an automatic and efficient system for information exchange between Strategic Computing personnel and the research community-at-large; *Wideband Communications*: exploration of the technology needed to support satellite-based wideband computer communication with dynamic intercommunication of many ground stations via packet-switched sharing of a single satellite environment; *CONSUL*: development of a knowledge-based interactive system that can provide services and explanations in response to user requests, including natural language requests from users with little computing experience; *CUE*: development of a methodology to automate the process of generating a consistent command-, editor-, and menu-based interface to integrated software applications using a knowledge-based approach: *Knowledge Delivery*: development of new methods for autonomous creation of text by machine, with the focus on fluent, easily controlled sentence and paragraph production; *Computer Research Support*: operation of reliable computing facilities and continuing development of advanced support equipment; *Exportable Server System*: providing high-reliablity services to the workstations on the local area network, in association with the Exportable Workstation Systems project: *Exportable Workstation Systems*: development of a remote testbed environment of advanced workstations and servers; *TCP/IP Implementation Support*: implementing TCP/IP protocols into active use in the operational ARPANET and MILNET by installing and debugging host software to support the ISI user community; *Interlisp*: creating a self-sustaining support center which provides large-address-space portable versions of Interlisp (ISI-Interlisp) for the VAX and for other hardware architectures; *New Computing Environment*: exploring, determining, and implementing the next generation of computers and computing facilities for the ISI research environment: *Strategic Computing - Development Systems*: providing development computers for the DARPA Strategic Computing program, system integration as required, and distribution mechanisms for disseminating the systems to the program participants; *Strategic C3 System Experiment Support*: participating in a Strategic Command, Control, and Communication systems experiment demonstrating and evaluating the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-based techniques).

ISI/SR-85-150

University of Southern California

1984 ANNUAL TECHNICAL REPORT

July 1983 - June 1984

A Research Program in Computer Technology

Principal Investigator and Executive Director:
Keith W. Uncapher

Prepared for the Defense Advanced Research Projects Agency
Effective date of contract 1 July 1981
Contract expiration date 31 May 1987
Contract # MDA 903 81 C 0335
ARPA Order 4242

INFORMATION SCIENCES INSTITUTE ISI
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by contracts with DARPA (MDA903 81 C 0335), NSF (MCS-8304190), and AFOSR (F49620-84-C-0100). Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, NSF, AFOSR, the U.S. Government, or any person or agency connected with them.

PROJECT FUNDING

The following projects were funded by the Defense Advanced Research Projects Agency, under contract no. MDA903 81 C 0335: Explainable Expert Systems, Information Management. Mappings, VLSI, Advanced VLSI, Command Graphics, Internet Concepts Research. Project NeaT, Strategic Computing Information System, Wideband Communications, Consul, CUE, Computer Research Support. Exportable Server System, Exportable Workstation Systems. TCP/IP Implementation Support, New Computing Environment, Strategic Computing - Development Systems, and Strategic C3 System Experiment Support.

The Knowledge Delivery Research project was funded by the Air Force Office of Scientific Research, under contract no. F49620-84-C-0100.

The Transformation-Based Maintenance project was funded by the National Science Foundation, under contract no. MCS-8304190.

The Interlisp project was internally funded by the University of Southern California.

CONTENTS

SUMMARY

This report summarizes the research performed by USC/Information Sciences Institute from July 1. 1983, to June 30, 1984, for the Defense Advanced Research Projects Agency, the Air Force Office of Scientific Research, and the National Science Foundation. The research is focused on the development of computer science and technology, which is expected to have a high DoD/military impact.

The ISI program consists of 22 research areas:

Explainable Expert Systems: creating a framework for building expert systems that enhances an expert system's explanatory capablities (allowing it to explain and justify its reasoning), and eases modification and maintenance of an expert system; *Information Management*: combining artificial intelligence, software technology, and database techniques to build a more highly integrated. uniform, and user-evolvable computing environment for both software development and information services; *Mappings*: development of transformations for converting high-level specifications of programs into implementations in software or VLSI; *Transformation-Based Maintenance*: supporting reimplementations of a framework that allows a programmer to formally develop software from specifications via transformations by automatically supplying sequences of low-level transformations which accomplish or facilitate high-level transformations: *VLSI*: providing a low-cost, fast turnaround LSI/VLSI device fabrication service to support a geographically distributed VLSI research community with no direct access to VLSI fabrication but with access to a computer communication network, and conducting research on the VLSI design problem; *Advanced VLSI*: providing access to 1.2 micron CMOS/Bulk fabrication, printed circuit board fabrication, and centralized functional testing of custom-designed parts prior to their assembly into packages and boards; *Command Graphics*: development of a device-independent graphics system and graphics-oriented command and control applications programs; *Internet Concepts Research*: exploring aspects of protocols for the interconnections of computer communication networks, specifically the design and implementation of an internetwork computer message system and the design of internetwork host and gateway protocols: *Project NeaT*: development of an advanced, information-oriented man-machine interface for aircraft, which will behave as an intelligent associate to the pilot—the interface will be based on an expert system, which models the decisions of an experienced copilot; *Strategic Computing Information System*: providing an automatic and efficient system for information exchange between Strategic Computing personnel and the research community-at-large: *Wideband Communications*: exploration of the technology needed to support satellite-based wideband computer communication with dynamic intercommunication of many ground stations via packet-switched sharing of a single satellite environment; *CONSUL*: development of a knowledge-based interactive system that can provide services and explanations in response to user requests, including natural language requests from users with little computing experience; *CUE*: development of a methodology to automate the process of generating a consistent command-, editor-, and menu-based interface to integrated software applications using a knowledge-based approach; *Knowledge Delivery*: development of new methods for autonomous creation of text by machine. with the focus on fluent. easily controlled sentence and paragraph production; *Computer Research Support*: operation of reliable computing facilities and continuing development of advanced support equipment: *Exportable Server System*: providing high-reliablity services to the workstations on the local area network. in association with the Exportable Workstation Systems project; *Exportable Workstation Systems*: development of a remote testbed environment of advanced workstations and servers; *TCP/IP Implementation Support*: implementing TCP/IP protocols into active use in the operational ARPANET and MILNET by installing and debugging host software to support the ISI user community; *Interlisp*: creating a self-sustaining

support center which provides large-address-space portable versions of Interlisp (ISI-Interlisp) for the VAX and for other hardware architectures; *New Computing Environment*: exploring, determining, and implementing the next generation of computers and computing facilities for the ISI research environment; *Strategic Computing - Development Systems*: providing development computers for the DARPA Strategic Computing program, system integration as required, and distribution mechanisms for disseminating the systems to the program participants: *Strategic C3 System Experiment Support*: participating in a Strategic Command, Control, and Communication systems experiment demonstrating and evaluating the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-based techniques).

EXECUTIVE OVERVIEW

USC/Information Sciences Institute (ISI) is a university-based basic and systems research and development center, with a diverse set of projects embracing both short-term and long-term DoD requirements. The key areas of ISI's research are software production technology, computer-based communication technology, VLSI, and the development of user-friendly computer interfaces. In addition, ISI remains a major computer support center for several thousand remote ARPANET and MILNET users.

New research efforts are addressing military mission requirements and the critical shortage of skilled technicians in the services. They include production of expert systems, computer-generated natural language, and fast-turnaround fabrication for VLSI designs.

Historically, ISI has had a major responsibility for technology transfer to the military. We continue to lead in this area because of our commitment to this vital process—it is a result of the increased need for military exposure to technology at an early stage of its development, as a precursor to shaping technology for specific military applications.

A highlight of this report is the growing dependence of hundreds of geographically distributed designers on MOSIS as a resource for fast-turnaround fabrication of simple and complex designs in a rich variety of implementation technologies.

At a time when the military base of this country is in critical need of technology—specifically developed for unique military requirements—the challenge to our flexibility, our system orientation, and our insight in anticipating military needs has never been greater. The research, development, and support capability outlined herein represents ISI's response to these challenges and to their opportunities.

The diversity of research interests at ISI provides a broad base for a healthy interchange and amplification of ideas and information, as reflected in the summaries below.

Explainable Expert Systems. The major goal of the Explainable Expert Systems (EES) project is to create a framework for building expert systems that enhances an expert system's explanatory capabilities, allowing it to explain and justify its reasoning, and eases modification and maintenance of an expert system. We use an automatic program writer to create the expert system from abstract knowledge of the domain. As it creates the system, the writer records its decisions in a development history. When the expert system is run, this development history provides explanation routines with the knowledge needed for good explanations.

Information Management. The Information Management (IM) project is constructing an operational testbed to support rapid prototyping of software, focusing on support of system designers and implementors. We are exploring gains from the use of *formal specifications* both as the starting point of software implementation and as the site of software maintenance. The research focuses on three objectives: producing software prototypes from formal specifications; evolving specifications; and integrating applications by sharing specifications.

Mappings. The Mappings project attempts to formally represent all crucial information used in the processes of specification, design, implementation, and maintenance. During the implementation phase of the program development cycle, the high-level constructs used to specify a program must be replaced by efficient realizations of these constructs. The Mappings project has concentrated on discovering and implementing the general transformations—or ***mappings***—required. We have found that specifications are often difficult to read, despite the high-level nature of the constructs used. The essence of the problem is that the modes of communication normally used between people are considerably richer than those between people and machines. We have several ideas on how this communication gap can be narrowed using incremental specification.

Transformation-Based Maintenance. The primary goal of the Supervisory Control of Transformational Implementation Systems (SCTI) project was to design and develop a framework for understanding, reusing, and maintaining previous specifications and optimizations. The SCTI project was a predecessor to the Transformation-Based Maintenance (TBM) project, which developed the system support needed to facilitate the automated implementation of specifications using transformations. The TBM project has extended the framework to deal with the activity of ***redesign*** of specification and implementation.

VLSI. The VLSI design commu[illegible]ies of DARPA and NSF require fabrication capabilities in order to investigate the design me[illegible]odologies and architectures appropriate to VLSI where gate-count will exceed one million gates per device. Recognizing this need, DARPA established the MOSIS (MOS Implementation Service) system at ISI in January 1981. MOSIS has reduced the cost of VLSI prototyping; shortened turnaround time for VLSI prototyping; freed designers from fabrication idiosyncrasies; and made design less dependent on specific fabrication lines. A large part of the MOSIS community is now designing in CMOS. In order to allow the community to build on top of the work it has already done, MOSIS will develop a library of community designs and make it available to new designers. MOSIS routinely supports nMOS at 3.0 and 4.0 micron feature sizes, with *buried*, rather than *butting*, contacts, in accordance with the Mead-Conway design rules. At least 20 vendors can fabricate devices according to these nMOS design rules. MOSIS supports CMOS/Bulk (typically P-well) with 3.0 micron feature size, usually with a capacitor layer and occasionally with a second metal layer. MOSIS supports CMOS/SOS fabrication with 4.0 micron feature size in accordance with the Caltech design rules. All of the SOS vendors support these design rules. Since its inception, MOSIS has used KL-2060s (running TOPS-20) not only to communicate with users on the network, but also for its considerable computing requirements. It has been determined that VAX 11/750s would provide a more efficient and economical environment for computing. Substantial progress has been made in porting the MOSIS geometry software to the VAX 11/750. MOSIS now generates MEBES files on the VAX 11/750. The transition to VAX 11/750s has been be transparent to MOSIS users; they continue to access MOSIS on KL-2060s. The VAX 11/750 will be used primarily for geometry crunching, preparing MEBES files, and writing MEBES tapes. The MOSIS vendor base has expanded substantially during this reporting period. Increased user feedback and more extensive test results have allowed the MOSIS project to determine and communicate fabrication requirements to new vendors. This has resulted in higher quality wafers and the development of consistently reliable vendor sources for mask making and nMOS fabrication.

Advanced VLSI. The main technology on which systems are dependent is VLSI. The natural first steps for the Advanced VLSI project are the provision of interfaces to 1.2 micron CMOS bulk technology necessary for systems based on 100,000-transistor chips; access to fabrication of printed circuit boards (PCBs) containing both custom-designed and standard VLSI parts; and access to centralized functional testing of custom-designed parts prior to their assembly into packages and boards. It is these services which must be provided for system designers if they are to easily assemble prototype systems. CMOS bulk technology with a feature size of 1.2 microns is currently being developed in the research departments of a number of commercial organizations. It is not expected to be generally available until late 1985. In the normal course of events, one would not expect system designers to be able to take full advantage of the technology until about 1988. The work performed by this project is expected to shorten this normal course of events by a full three years. It allows the technology to develop in parallel with design rules and circuit and systems design techniques. The primary goals of the Advanced VLSI project are to provide the DARPA VLSI systems design community access to 1.2 micron CMOS bulk technology and to PCB fabrication; and to provide centralized functional testing of custom-designed parts prior to their assembly into packages and boards.

Command Graphics. The Command Graphics project builds upon previous work performed by ISI for DARPA to make graphics more readily available and to enhance decision making capability in the military command and control environment. In a previous contract, ISI developed a graphics system for use in command and control applications. Two important observations were made from the previous work. First, the production of legible geographic-based displays requires a considerable amount of graphics intelligence and sophistication. In fact, the amount of programming necessary to incorporate the "graphics intelligence" often exceeds that of the application program itself. Second, the representation mechanisms used in Situation Display were essentially generic in nature. During the reporting period, substantial effort was directed toward producing the intelligent agent described above, hereafter referred to as the Geographic Display Agent (GDA). The architecture of the overall system was defined, the programmer interface to the system was specified, a design effort to identify and relate specific system components was undertaken and completed, and a partial implementation of the GDA was completed. This version of the GDA allowed the application to specify a desired map area and place icons with associated textual annotations on that background map. The GDA provides the C2 application programmer with an object-oriented interface that automates many of the functions a cartographer normally performs when creating geographically oriented displays. Its purpose is to allow the application programmer to concentrate on deciding which objects to display rather than the mechanics of actually generating an aesthetic display.

Internetwork Concepts Research. The goal of the Internet Concepts Research project is to extend and enhance computer communications. The project's work is based on the ARPA-Internet, a system of interconnected computer communication packet networks. The rules of communication (or protocols) are the key element in successful computer communication. The ARPA-Internet has working protocols for communication between heterogeneous computers via an interconnected collection of heterogeneous packet networks. The ARPA-Internet currently has 933 hosts, 158 networks, and 89 gateways. This research covers work at several levels, involving applications protocols as well as host-to-host and gateway-to-gateway protocols. In the applications level protocols, the focus is on new uses of the Internet based on new procedures for abstract process communication, hierarchical naming, multimedia mail, high-speed network interfaces, and new capabilities in the gateway protocols. In the gateway and host level protocols, these extensions include mechanisms and procedures for monitoring and maintaining performance and survivability, for allowing continued growth of the Internet, and for dynamic control of the Internet. The basic protocols are largely complete, but a number of extensions are being explored to integrate and extend the packet network technology. The long-term goals of the Internet Concepts Research project are to provide appropriate and effective designs for the primary user service applications in the internetwork communication environment. The designs are based on a set of host and gateway level protocols that provide the full range of service characteristics appropriate to a wide variety of applications. Our approach has been to pursue in parallel the abstract design and experimental implementation of protocols. The interaction of these activities provides valuable insights into problems and potential solutions. Further, we divide our effort into several task areas: Abstract

Process Communication. Hierarchical Naming System, Multimedia Mail, High-speed Interfaces, and Studies, Surveys, and Specifications.

Project NeaT. The NeaT project is studying the problem of information transfer from sophisticated aircraft systems to pilots. To accomplish this research task, the project will design an advanced, information-oriented man-machine interface for aircraft, which will behave as an intelligent associate to the pilot. The interface designed by the NeaT project will be applicable to a wide range of aircraft: military and civilian, light and heavy, combat and transport. The interface will be based on an expert system, which models the decisions of an experienced copilot. The system will be knowledgeable in areas such as fuel management, navigation, and aircraft reconfiguration in response to damage or emergency. A major focus of the project will be the interface between this expert system and the pilot: the project will study various methods and philosophies of information presentation, with an emphasis on intelligent speech and modifiable visual displays. Besides building a working generic system, this research will produce a body of knowledge that future researchers may build upon.

Strategic Computing Information System. The success of the Department of Defense's Strategic Computing research program depends on the easy dissemination of program-related information to a geographically distributed research community. The Strategic Computing Information System (SCIS) project provides an automatic and efficient system for information exchange between the Strategic Computing personnel and the research community-at-large. Initially, a basic SCIS will be implemented on TOPS-20, using modified code originally developed for MOSIS. Once the SCIS is in operation, it will be extended to IBM-PC and VAX implementations. These later implementations of the SCIS are expected to be more complicated, allowing us to incorporate the lessons we have learned from the first use on the TOPS-20 system. The receipients will be those who can send and receive electronic mail to the SCIS. Initially, this group will include the ARPA-Internet mail community (*ARPANET, MILNET, CSNET, DEC,* Xerox, etc.) and the TeleMail users; soon, it will include MCI Mail (including TELEX) and TYMNET users. Typical transactions of the Strategic Computing Information System will be announcements; requests for information/information retrieval; Commerce Business Daily/Request for Proposal-related activities; information dissemination: queries; and administrative requests (join, address change, etc.). All such communication will be handled without exception via electronic mail (ARPANET, TeleMail, MCI Mail, TYMNET).

Wideband Communications. The Wideband Communications (WBC) project is one of several groups involved in the joint DARPA/DCA Wideband Packet Satellite Program. The objective of the Wideband Program is to explore the technical and economic feasibility of packet voice on a large scale, and to investigate other media, such as packet video. ISI's role is to conduct experiments using the Wideband Network as it becomes available for regular, active use. ISI has been closely involved in efforts to make the network as reliable and useful as possible, and has established a schedule of regular tests and experiments in an effort to encourage real use of the network, thus gaining as much practical experience as possible. The overall approach of the ISI WBC project is twofold: to expedite the transition of mature technology, such as packet voice, from the laboratory into the everyday working environment; and to apply the unique capabilities of the Wideband Network to new areas, such as packet video and high-rate data transmission. This approach requires a broad spectrum of efforts, including development of hardware, software, and protocols.

Consul. The Consul project is exploring the use of knowledge-based technology to allow a natural interface between users and computer services. We conduct our research in the framework of a demonstrable user interface environment that includes facilities for understanding natural language requests and producing natural language help and explanations. The system contains detailed models of users and services and a set of inference mechanisms to transform descriptions from one model into an appropriate description in the other. Consul research therefore focuses on representation and inference in the interactive systems domain. Consul's interface is designed to be used in conjunction with CUE, which provides a menu/command-based interface that interacts with Consul's knowledge-based components. The interface is provided in a distributed environment. with Consul residing on a server machine that can be shared by one or more CUE workstations on the same Ethernet.

CUE. The goal of the CUE project. now discontinued. has been to provide an extensible environment for building and using integrated interactive computer services. In CUE, there are no boundaries between, say, the electronic mail service and the automatic calendar service. This type of automatic inter-service interaction requires an environment in which individual services can make assumptions about the properties and behavior of the other services in the system. These assumptions must relate to data structures, functional capability, and state information. The objective of the CUE project was to produce such a consistent underlying environment (CUE), in which users can interact with the system without regard to service boundaries. Although CUE has been discontinued, the ideas explored by CUE are being used to form the basis for the new Formalized System Development project.

Knowledge Delivery Research. The general practical goal of the Knowledge Delivery project is to allow computers to express information in appropriate, easily read, multiparagraph English text. The project's text generation system is called Penman. Parts of Penman are programmed, and other parts are at various levels of specification as program designs. One of Penman's modules is Nigel. the grammar. Nigel has a more explicit and formal notation for structure building than any predecessor. We have created a new body of theory. called *Thetorical Structure Theory* (RST). which represents the structure of written monologues such as reports, magazine articles. intructions. and other expository kinds of text. In follow-on projects already under way, we will distribute the Nigel grammar to other research organizations and will build several text generators for specific purposes. We will also work in the areas of lexical selection. noun phrase planning, sentence planning. interfacing Nigel to various programmed knowledge notations, text planning, text brevity techniques. and appropriate representation of the static and dynamic knowledge of the reader.

Computer Research Support. The Computer Research Support project is responsible for providing reliable computing facilities on a 24-hour, 7-day schedule to the ARPANET/MILNET research and development community. At the same time, the project makes available to ARPANET/MILNET users the latest upgrades and revisions of hardware and software. The project provides continuous computer center supervision and operation. and a full-time customer-service staff that is responsive to user inquiries. This project supports two computer installations: the larger, main facility at ISI in Marina del Rey, and the Design Center at Gunter Air Force Base in Alabama. The ISI Information Processing Center provides support in four tightly interrelated, though distinct. areas: Hardware, System Software, Operations. and Network Services. The overall charter of the organization is to assure that the needs of the user community be addressed and protected as effectively as possible. All groups are concerned with the efficient use of the machines and with the security of the physical plant and all online information.

Exportable Server System. The Exportable Server System project is associated with the Exportable Workstation Systems (EWS) project. Together, these efforts seek to develop and provide a sophisticated information system for the 1990s. A principal goal of the Exportable Server System is to provide services of high reliability to the workstations on the local area network. It will probably not be economically feasible for onsite technicians to be available for most environments of this type: thus there is a need for highly reliable or redundant services which can be controlled from a remote service center.

Exportable Workstation Systems. The Exportable Workstation Systems project is associated with the Exportable Server System (ESS) project. While the principal goal of this project is to install a workstation environment at a remote location (the DARPA-IPTO offices in Arlington, Virginia), it is also intended to provide a testbed for several other concepts and projects. The environment will be configured to accommodate a variety of workstation models with varying capabilities. This configuration will provide a minimum set of services for each workstation model: document preparation, mail handling, file storage and manipulation, and administrative activities (spreadsheets, calendars, etc.). These services will be provided either directly on the workstations, or on the central server (a DEC VAX-11/750). All of the workstations will be connected to a local Ethernet and to the ARPA-Internet via a BBN-maintained gateway and by equivalent code on the VAX that will function as a gateway in the event of failure.

TCP/IP Implementation Support. The Department of Defense has adopted the Internet concept, and the IP and TCP protocols in particular, as DoD-wide standards for all DoD packet networks. The DoD will be converting to this architecture over the next several years. The role of the TCP/IP Implementation Support project is to assist in placing these protocols into active use in the operational ARPANET and MILNET by installing and debugging host software to support the ISI user community. This effort involves programmers and researchers from ISI, Bolt Beranek and Newman, Inc., Digital Equipment Corporation, UC Berkeley, SRI International, Stanford, the Massachusetts Institute of Technology, and other institutions.

Interlisp. This project created a self-sustaining support center for Interlisp. The project provides large-address-space portable versions of Interlisp for the VAX and for other hardware architectures. Currently, ISI-Interlisp is implemented on the VAX and runs under the VMS and UNIX operating systems. In addition to ongoing maintenance tasks, development efforts are planned for the future, including the porting of Interlisp to other machines.

New Computing Environment. The New Computing Environment (NCE) project's goal is to to adapt developing computer technologies to serve the research and military user communities. The resulting model computing environment will serve several purposes. It will provide a very large improvement in languages, system support, and additional computing cycles to the ongoing ISI research effort; it will serve as a testbed and eventual existence proof for the implemented technology; and it will serve as a proving ground for local computer environments targeted for DARPA and the military community, as well as a device for investigating new research ideas that will eventually be adapted by those communities. In addition, the small size, portability, and local computing capability of personal computers will allow for experimentation and realization of command and control requirements for an environment that can exist in mobile command centers.

Strategic Computing - Development Systems. This project will provide development computers for the DARPA Strategic Computing program, system integration as required, and distribution mechanisms for disseminating the systems to the program participants. In addition, the project will define an architecture for system communications and resource sharing among the computers acquired.

Strategic C3 System Experiment Support. DARPA has defined an experiment in Strategic C3 systems to be conducted in cooperation with the World Wide Military Command Control System (WWMCCS) System Engineer (WSE) and the Strategic Air Command (SAC). The concept of the experiment is to demonstrate and evaluate the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-base techniques) for strategic command, control, and communication. ISI provides an initial core of necessary facilities (ARPANET/MILNET access, host systems, various software tools, Network Services support, etc.) to allow SAC personnel to gain experience with this technology and to ensure the success of the experiment. Installation of modems, installation of 30 or more interactive CRT terminals, user training, system software training and support, and on-site maintenance of equipment are part of the continuing program.

SOFTWARE SCIENCES DIVISION

The Software Sciences division is attempting to provide order-of-magnitude improvements in software development and maintenance by automating the processes involved. This necessitates formalizing these processes and making major changes in the software lifecycle. In particular, we believe that the basic cycle should be formal specification followed by mechanical derivation of an efficient implementation.

By mechanically deriving the implementation from the formal specification, the resulting implementation is guaranteed to be consistent with the specification. Its efficiency is dependent on the appropriateness of the optimizations selected. In general, automating these selections, especially at the strategic level, is beyond the state of the art. Therefore, in many instances, we expect this to be a human-guided process. This derivation (and any human guidance provided) is recorded to enable reimplementation during maintenance.

Unfortunately, formal specifications are hard to write—partly because of inadequacies in specification languages, partly because more knowledge (information) must be made explicit, but also because the systems being specified are dynamic and anticipating all possible interactions between the component parts is difficult, if not impossible. It is therefore essential to ensure that the formal specification captures the user's intent. This can best be done through treating the formal specification as a prototype which can be tested to provide feedback on the behavior generated and insight into improving the specification.

In this new software paradigm, maintenance is performed on the formal specification rather than the implementation. To the extent that the specification describes *what* to do rather than *how* to do it, modification is greatly simplified because information is still localized; the optimization process spreads information and increases the coupling and implicit assumptions between parts of the system. A new implementation can then be generated from the revised specification either by rederiving it, or by replaying the recorded development, possibly modifying some of the optimization decisions previously made.

While we are interested in all aspects of the software process mentioned above, we necessarily must limit our objectives at any point in time. We started by focusing on helping users create formal specifications. We built a tool which translated informal natural language specifications into formal ones. Recognition of the inadequacies of the existing formal specification languages led to our development of a new class of specification language (Gist), based on formalization of the informal constructs used in natural language specifications. We developed natural language explanation capabilities which paraphrase the specification and characterize its behavior to improve user comprehension. We also developed an interactive program transformation system and a formal representation of the transformation-based derivation suitable for replay.

Our current objectives are a more principled development of systems, techniques to support evolution of such systems, and an operational rapid-prototyping facility. Each of these objectives is the central focus of a project within the Software Sciences division. The Explainable Expert Systems project is exploring the more principled development of expert systems through use of a structured model of the application domain, expertise in that domain, and strategies for using that expertise.

The Mappings project is extending the notion of specification to allow incremental elaboration of a specification from an idealized kernel. Finally, the Formalized System Development project is developing an operational rapid prototyping facility. Each of these projects is described in more detail in the following chapters.

1. EXPLAINABLE EXPERT SYSTEMS

Research Staff:
William Swartout
Robert Neches

Research Assistants:
Johanna Moore

Support Staff:
Audree Beal

1.1 PROBLEM BEING SOLVED

Most current expert systems provide implicit expertise. Although they can often perform at expert or near-expert levels, their expertise is not supported by a solid understanding of their domain. The shallowness of their expertise is revealed by some important limitations:

- ***They have limited (or no) explanatory capability.*** To be acceptable to users, expert systems must be able to explain their reasoning—not only what they do, but also why what they do is reasonable. Unfortunately, most expert systems are limited to "cookbook" explanations that state what the system did; they cannot justify their actions because the rationale behind the system's rules or methods is not represented. Additionally, it is often difficult to understand the expert system's overall strategy because high-level strategies are implicitly encoded in low-level rules.
- ***They are difficult to modify or extend.*** The implicit encoding of a general heuristic or principle in many highly specific rules also hampers modification of the system. If the general heuristic needs to be modified, the system builder must manually locate and carefully modify each of the specific rules that encode it.
- ***They cannot be easily adapted to new domains—even those closely related to the original domain.*** Again, the problem is that the limited representational power of current frameworks forces almost all of the rules or methods of the system to be highly domain-specific.

These limitations stem from two fundamental problems. First, current frameworks encourage system builders to write rules or methods in a highly "compiled" style. The rules or methods incorporate many different kinds of knowledge (such as domain facts, heuristics for achieving goals, and efficiency concerns). They should be represented separately and explicitly, but they never are. This intertwining of knowledge hampers modification and extension. Second, the process of compiling general domain facts and problem-solving knowledge into specific rules is performed, unrecorded, in the system-builder's head. The design decisions behind the system are not available; hence, machine-produced justifications of its behavior cannot be provided. Problems in explanation, maintenance, and extension can all be traced to the same underlying causes: the lack of a record of the system's development, and the use of a low-level, relatively compiled representation for rules or methods.

The major goal of the Explainable Expert Systems (EES) project is to create a framework for building expert systems that:

- enhances an expert system's explanatory capabilities, allowing it to explain and justify its reasoning, and
- eases modification and maintenance of an expert system.

1.2 GOALS AND APPROACH

The creator of an expert system can usually provide excellent explanations of how it works. Our approach extends the paradigm developed in the XPLAIN system [3]. We use an automatic program writer to create the expert system from abstract knowledge of the domain. As it creates the system, the writer records its decisions in a development history. When the expert system is run, this development history provides explanation routines with the knowledge needed for good explanations. A broad view of the EES system is shown in Figure 1-1.

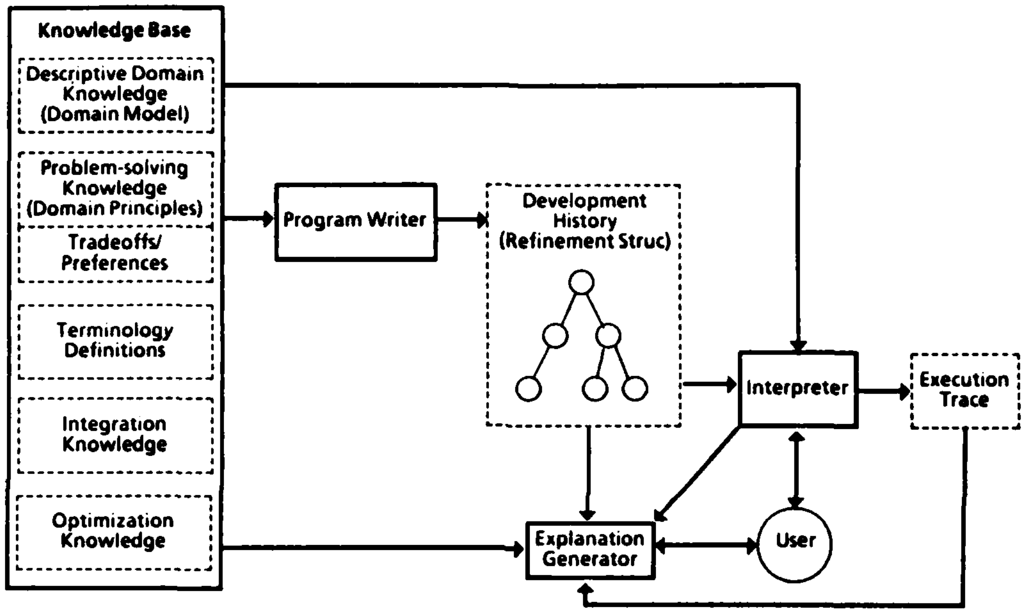


Figure 1-1: Global view of the EES framework

To address the problem of too-specific rules, we provide the system builder with a more expressive knowledge base that encourages him to represent knowledge more abstractly and to explicitly separate the different kinds of knowledge that are part of an expert system. The *domain model* describes how the domain works. It contains typological and causal linkages, among other things. However, it does not indicate how problem solving should be done. *Domain principles* represent problem-solving strategies and are used by the program writer to drive the refinement process. *Tradeoffs* are associated with domain principles to indicate the beneficial and harmful effects of selecting a particular strategy to achieve a goal. *Preferences* are associated with goals and are used to set priorities based on tradeoffs. Mappings between abstract terms and the concepts that realize them (which had been included as part of domain principles in XPLAIN), are separated as types of knowledge, allowing *terminology* to be shared across domain principles. *Integration knowledge* is used to resolve potential conflicts among knowledge sources. *Optimization knowledge* represents ways of efficiently controlling the execution of the derived expert system.

The EES framework will generate a runnable expert system by applying the *program writer* component to the knowledge base. The steps taken in producing code are recorded in a *development history*, a lattice structure whose leaf nodes represent system implementation code and whose interior nodes represent goals and decisions made on the way to generating the implementation. The *interpreter* executes the leaf nodes of the development history. It produces an *execution trace*, and manages the system's normal interaction with users. User questions, however, are sent to the *explanation generator*, which accesses the knowledge base, development history, interpreter, and execution trace in the course of constructing answers to queries.

1.3 SCIENTIFIC PROGRESS

The EES framework is approaching the implementation stage. An expert system called the Program Enhancement Advisor has been selected as a testbed domain for EES. This system is intended to heuristically critique a user's LISP program, suggesting simple enhancements that could be made to the program to improve its readability, efficiency, or maintainability.

Using programs provided by research programmers in our laboratory as input, and relying on both our own LISP knowledge and the expertise of two highly skilled builders and maintainers of LISP-based systems, we have identified approximately a dozen enhancements that the program ought to be able to perform. Interviews with our domain experts and analysis of their "thinking-out-loud" protocols given as they rewrote the example programs were used to identify useful enhancements, and to attempt to determine the principles underlying them. At the same time, we have investigated the kinds of explanations needed in expert systems and the knowledge representations required to support them. As a result, we have identified the knowledge base components that are required to support 13 classes of possible questions.

As can be expected in an approach that relies heavily on detailed knowledge representation, much of the preliminary work is in identifying key general concepts in the knowledge domain and determining how to model those concepts in the representation language. Most of those policy decisions have now been made. We have developed an initial version of a language for expressing goals and plans, and have established ground rules for representing concepts such as transformations, syntactic structure of programming constructs, semantic components of programming constructs, and some aspects of the effects of evaluating an expression. Using those ground rules, we have completed the process of encoding the domain and problem-solving knowledge related to one transformation in NIKL [2], the representation language being used. We are in the process of encoding the knowledge pertaining to three other transformations.

Design of the program writer is nearing completion. We expect to have a working version of the program writer in late 1984. Explanation strategies have been devised for 6 of our 13 question types, but none have yet been implemented. To create one of the critical components of the explanation generator, an English paraphraser of NIKL knowledge structures, we are generalizing an existing NIKL paraphraser [1] to employ more principled methods for producing natural language.

1.4 IMPACT

We expect the EES project to provide a new paradigm for developing expert systems that will:

- provide better explanations, because the rationale behind the system will be captured;
- be easier to extend and modify, because they will be more modular;
- be more sound, because their development will be more principled.

1.5 FUTURE WORK

A prototype version of the EES framework will be complete early in 1985. Shortly after that, we will complete an initial version of the Program Enhancement Advisor. During 1985 we will design and construct an explanatory facility for EES that will allow expert systems constructed using the EES framework to be explained and justified in English. We will test our claim that the EES framework eases the maintenance of an expert system by using it to extend the coverage and enhance the scope of the Program Enhancement Advisor. We will also investigate the feasibility of having the automatic programmer create programs which contain assumptions that increase their efficiency. Because these assumptions and the deeper, assumption-free knowledge behind them will be explicitly represented, we hope to create expert systems that are both efficient and robust. They will handle simple cases with compiled, assumption-based reasoning, reverting to deep knowledge for complex cases. Toward the end of 1985, we will test the generality of the EES framework by using it to implement another expert system in a different domain.

REFERENCES

1. Lipkis, T., and W. Mark, Knowledge Base Explainer Design Document. USC/Information Sciences Institute, CONSUL Project Internal Design Note.

2. Moser, M. G., "An overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding*, Bolt Beranek and Newman, Inc., Cambridge, Mass., 1983. BBN Technical Report 5421.

3. Swartout, W., "XPLAIN: A system for creating and explaining expert consulting systems," *Artificial Intelligence* 21, (3), September 1983, 285–325. Also available as USC/Information Sciences Institute, RS–83–4.

2. INFORMATION MANAGEMENT

Research Staff:	*Research Assistants*:	*Support Staff*:
Neil Goldman	Brigham Bell	Audree Beal
Robert Balzer	Bill Betts	
Don Cohen	Seth Goldman	
Dave Dyer		
Matthew Morgenstern		
Robert Neches		

2.1 PROBLEM BEING SOLVED

The Information Management (IM) project is constructing an operational testbed to support rapid prototyping of software. focusing on support of system designers and implementors. We are exploring gains from the use of *formal specifications* both as the starting point of software implementation and as the site of software maintenance. The research focuses on three objectives:

1. producing software prototypes from formal specifications;
2. evolving specifications;
3. integrating applications by sharing specifications.

First. we want to use formal specifications as prototypes of the specified systems so that insight gained from feedback of their behavior can be used to improve those specifications before they are implemented. To support this. the testbed must support rapid (relative to traditional implementation technology) production of operational software from specifications.

Second, given a discrepancy between the formal specification and intent. or insight into an improvement, the testbed should facilitate changing the specification to remove the discrepancy or incorporate the improvement.

If the first two objectives can be met. this specify-test-evolve cycle will be repeated many times to incrementally derive a satisfactory system, replacing the one-shot "batch" approach used today. One of the key insights gained by deriving implementations from formal specifications will be that tight integration of systems is a natural by-product of designing at a specification—rather than implementation—level. Thus. our final objective is support for the tight integration of diverse "services" into a single computing environment. in contrast with the unintegrated collection of applications that characterize current computing environments.

2.2 GOALS AND APPROACH

The IM project will demonstrate how certain software technologies developed at ISI and elsewhere over the last five to ten years can be consolidated into a single programming and computing environment. The IM environment is superior to previous environments in three primary areas: software production. software maintenance. and end-user software maintainability.

2.2.1 Software Production

Software is developed in IM by the writing of an operational specification of the desired software behavior and its transformation into an executable LISP program. Initially, IM is using an extension of Zetalisp [4] as its specification language, which we will refer to as EZL. The transformations that implement EZL specifications are implemented as macros and are applied automatically by the language compiler/interpreter. In the present configuration, a relatively high price is paid in terms of implementation quality in using the specificational extensions available in EZL. Our next specification language will continue to be implementable by transformations applied automatically by the compiler/interpreter, but it will consider optional implementer-provided *annotations* in order to select among alternative transformations [3]. This configuration will greatly justify the price paid for using many of the specification constructs. The final quantum change will be to a specification language which, like Gist [1], cannot in all cases be automatically transformed into an implementation. It will be necessary for an implementer to provide a sufficient body of annotations to direct the transformation of the specification into an implementation. The expected environment for producing implementations will permit the implementer to select the transformations *interactively* [2], exploring the space of possible implementations to find an acceptable one.

2.2.2 Software Maintenance

In each of the three increasingly sophisticated modes of software production described above, the specification, together with the record of transformations applied (whether those transformations were selected automatically, indicated by annotations, or selected interactively), provides a sufficient basis for regenerating the actual implementation code. This is the key to improving the technology for software maintenance. Maintenance will be performed by altering the specification, not the final implementation. In most cases, it is far simpler to express the desired alterations at the level provided by the specification. The transformation record of the specification's prior version implementation will then be used to greatly automate the activity of generating an implementation of the altered specification [5].

2.2.3 End-User Software Extensibility

End-user software extensions are not formally different from maintenance changes. However, only changes that meet certain conditions are reasonable for users other than system maintainers to make.

- *Locality of effect.* The user must be able to change a specification in desired ways without knowledge of the full specification or the details of its implementation.
- *Independence of expression.* It is highly desirable that specifications of user extensions or modifications be maintainable in a form that is separate from the "standard" service definition, rather than as a modified copy of some part of that specification.
- *Automatic implementability.* It must be possible to implement the user's extension automatically, without requiring him to provide annotations. It is acceptable if the syntax for specifying extensions goes beyond what can be correctly implemented automatically, provided that there is a guaranteed means of detecting when this is the case.

The primary IM specification mechanism having these characteristics is the *automation rule*. It provides a means for *autonomous* specification of an action to be taken whenever an indicated condition arises, where the condition is an arbitrary predicate on the database. The effect of the rule is localized in the data space of the system, although it may be distributed in the control space. We believe that its implementability can be automatically determined from the transformation history of the specification being extended.[1]

2.2.4 Testbed IM System

The IM project will construct a testbed IM system populated with several useful services. The testbed will be will be used daily by members of the project. It will serve three primary roles by providing the following:

1. a means of demonstrating progress to individuals outside the project.
2. feedback from its regular users on the value and limitations of the mechanisms it incorporates.
3. an experimental framework for measuring the viability of proposed mechanisms relative to the bounds imposed by compilation technology and the implementation hardware capacity.

In addition, the services that populate the testbed will provide functionality useful to the researchers in their daily work, some of which is not otherwise available on the supporting hardware. We expect that the IM programming environment will soon be clearly superior to other available LISP programming environments.

The initial hardware base for IM consists of dedicated Symbolics model 3600 workstations. They are sufficiently powerful LISP processors for the initial testbed prototype: they could be constructed and used with relatively little investment in performance tuning. We plan to port the system to less expensive LISP workstations, with the option of running IM as a self-contained system on the workstation or as a system distributed between a workstation front-end managing the user interface and a more powerful LISP machine acting as a "server" for this front-end.

2.3 SCIENTIFIC PROGRESS

In large part, this reporting period has been devoted to constructing the first version of our IM testbed environment and to gaining experience with it. Our efforts have been divided into four areas: the kernel computing environment, the programming environment, specific computing services, and the user interface.

2.3.1 Computing Environment

A number of changes have been made during the past year to the computing environment of IM, and designs for further improvements have been made. The focus of this work is on enhancing the *specificational* aspects of IM.

2.3.1.1 Virtual database

The existing implementation of IM is based on AP3, a language that extends LISP to provide access to a global database of relationships over typed objects. AP3 provides a uniform representation and indexing scheme for all relationships in this database, independent of the access paths to the data

[1]This is definitely the case for the first two stages of specification language outlined above. The degree to which useful extensions can be automatically implemented in a system that was interactively transformed is an open question.

that are actually used by an application program. This scheme often places a heavy execution time and space burden on an application, sometimes necessitating the retention of a second representation of the data, optimized for the indiosyncratic mode in which it is accessed. When this is done, it then becomes necessary to add further mechanisms for ensuring consistency betweeen the two representations.

Work has been done in preparation for replacing AP3 with a new *virtual* database foundation for IM. Under the new scheme, the declaration of a relation will be annotated with a specification of its implementation class. Associated with this implementation class is the information necessary to compile insertions, deletions, tests, and generations of tuples from the relation expressed in an implementation-independent syntax. Several implementation classes are predefined, but the set is open-ended: to add a new class one has only to define the protocol for (a subset of) the four accessors just mentioned. Programs are written in terms of these accessors and are compiled with respect to the implementation annotations. The annotations may include effort estimates, which the compiler uses in cases where alternative algorithms are available for generating answers to a retrieval (e.g., to generate functions that CALL a given function F1 and are CALLED-BY a given function F2, the compiler will use any effort annotations to decide between using CALL to generate and CALLED-BY to filter, or vice versa).

The virtual database will also support several concepts missing from or only partially supported by AP3. This support should improve both the expressiveness of IM service specifications and the efficiency of their implementations:

- *Computed relations* are relations that support neither assertion nor deletion of tuples. A typical use of such relations is to interface to standard run-time execution facilities of the host computing environment, such as comparing integers or times, or retrieving file attributes.
- *Derived relations* are relations whose testing and generating implemenatations are derived from a formula of first-order logic, e.g., BACKUP-PHONE(employee, p#) = = Es.SECRETARY(employee,s) and PRIMARY-PHONE(s,p#). Particular derived relations may be given insertion and deletion implementations that translate these forms of access into insertions or deletions of the relations from which they are derived.
- *Types* map into unary relations (but will still have special status).
- *Well-formed formulae* of first-order logic will be used throughout the syntax as the language of predicates—in particular, as the triggering patterns of rules. Rules for the scope of pattern variables are the standard rules for quantifier scope.

2.3.1.2 Persistent objects

Because both AP3 and the newly designed virtual database represent their data in the LISP virtual address space, some additional capability is needed to record and restore any information that needs to persist across new incarnations of an IM environment. Traditional operating systems provide a file system as a repository for such information; they expect each application to provide its own mapping of persistent information between virtual address space and file representation. IM services can use the host environment's file system for this purpose, but we would like to provide more assistance than this to service builders.

A batch save/restore implementation has been in use in our IM testbed since early 1984. It views the database as a network of objects (nodes) linked by attributes (arcs). It permits the specification of a subnetwork to be saved on a file. The subnetwork is specified via *descriptions* to specify nodes, and *views* to specify arcs. The mapping from a subnetwork in virtual address space to a file representation is handled automatically. Restoration of a subnetwork from a file requires overlaying that subnetwork on the database present in the restoring environment, and has the effect of adding new objects and/or attribute links to that environment.

While this batch implementation has proved useful, several deficiencies have led us to design an incremental capability for saving and restoring, based on the recording of a persistent history of primitive operations on the database. This design has been partially implemented.

2.3.1.3 Rule-based computation

We have implemented in IM a two-tiered rule scheme. *Coordination* rules have an associated invariant predicate and are activated whenever a change to the database violates the invariant. *Automation* rules have an associated trigger predicate and are activated whenever that predicate becomes true. Coordination rules execute before automation rules, and may interrupt them.

We have designed a semantics for coordination rules that conforms more closely to their intended use. Under this semantics, a coordination rule cannot undo the effect of another coordination rule run in the same execution cycle, nor is it possible for the net effect of a set of coordination rules to be sensitive to the order in which they are chosen to run. This design has not yet been implemented.

2.3.2 Programming Environment

A programming environment has been implemented within the IM computing environment. EZL is the target language for programs constructed and maintained in this environment. The programming environment comprises two major facilities: the first facility aids in the organization and management of software systems; the second facility records the development of software systems.

In the standard Zetalisp programming environment, a system is managed as a collection of ASCII files, each of which is made up of a sequence of defining forms. In the IM programming environment for EZL, each individual definition corresponds to a distinct object whose type depends on the kind of thing being defined: function, structure, global variable, etc. The defining form is the value of this object's SourceText attribute. These objects can be hierarchically composed into modules; a given object may be part of multiple modules. Single definitions or entire modules may be installed in a user's execution environment. The programming environment records the definitions that have been changed since they were last installed. It also keeps track of who is responsible for maintaining and documenting each module.

Any module in this environment may be declared to constitute a *system*. A development history is maintained for each system. This history records a hierarchy of development goals, with actual creations of or modifications to individual definitions at the leaves of the hierarchy. The non-terminal nodes of this hierarchy have text attached to them (supplied by the system maintainer), explaining the purpose of each goal. Changes to individual definitions are time-stamped and attached to this goal hierarchy by automation rules which are triggered by the completion of edits to the defined objects (see Section 2.3.4.2). This record of past definitions makes it possible to revert to earlier versions of particular definitions, modules, or even the entire system.

2.3.3 Integrated Services

Several services were constructed to operate in the IM computing environment. They have been in daily use by researchers on the IM project staff, both for practical administrative purposes and as vehicles for testing the ideas embodied in the IM computing and programming environments.

2.3.3.1 Electronic mail

The *Electronic Mail* service (EM) allows IM users to receive ARPANET messages into their personal programming environments and to create and send ARPANET messages from their environments.[2] EM provides a menu-based interface to messages, including standard mail manipulating commands for READ, FORWARD, REPLY, and DELETE. Through the definition of a textual view of a message, which is simply the text of its body, the generic capabilities of EDIT and HARDCOPY are provided. There are some interesting differences between EM and traditional mail management programs (e.g., MM or HERMES):

- The use of mailing lists as addressees in outgoing messages has been subsumed by the ability to use *descriptions* as addressees. This means that the set of individuals to whom a message is sent can be specified such that it is computed at message-sending time on the basis of data in the user's own database.
- Many outgoing messages may be partially drafted and left in an incomplete state in the user's database. Because these messages are objects, they may be manipulated by the generic IM operations. They may thus be completed, sent, and/or deleted when the user desires.

2.3.3.2 Alarm clock

The generic capability to trigger program invocation from changes to the user's database provides a powerful mechanism for automating activities. The *Real-Time Clock* is a significant portion of the computing environment's state that changes without any reflection in the database. The *Alarm Clock* service (AC) allows IM users to schedule program invocation on the basis of this clock. The clock and the requests in its queue are objects in the database. A protocol is provided that permits a request to reschedule itself for reexecution at a later time.

2.3.3.3 ISI Employee Database

The ISI Employee Dabatase (EDB) contains workplace information about approximately 150 ISI employees, including their office locations, telephone extensions, research projects, and ARPANET mail addresses.

2.3.3.4 Interservice interaction

One of the primary goals of IM is to provide a computing environment in which services integrate naturally, without the need for prescience on the part of the original service designers. The primary integrating force in the IM design is the use of a uniformly accessed global database as the repository of all information. Several examples of natural integration have already been observed:

- ISI employees (EDB) as the referents of descriptions used as outgoing message addressees (EM).

[2] EM is entirely independent of the ZMAIL electronic mail service supplied by Symbolics Corp. on the 3600 model workstations.

- Incorporation of newly arrived electronic mail (EM) into the user's environment at regular time intervals (AC).
- Routing of bug reports about services to the maintainer of the service (EDB) via electronic mail (EM).

2.3.3.5 Service extension

One of the primary goals of IM is to permit relatively sophisiticated users of services to tailor and extend the services described above to meet their own personal needs, again without the need for prescience on the part of the original service designers. The primary forces supporting natural extensibility in the IM design are the provision for rule–based processing, and the ability to extend the domain model that drives a service. Several examples of natural extension to services by users have already been observed:

- Automatic classification of incoming messages into categories (subtypes) based on keyword matching with the message subject.
- Automatic display of incoming messages as graphic icons, with links from the message icons to category icons.
- Automatic response to messages when the addressee is away from his mail processing capabilities for an extended period.

2.3.4 User Interface

The user interface to IM consists of rectangular windows whose size, shape, and position on the bitmap display screen are independently controllable. There are three relevant windows:

- *LISP Interaction.* The LISP interaction window permits the user to type in EZL forms to the interpreter and have their values printed back in the window. This window is also the default area for side–effect printing, user prompting, and error messages. All objects printed to this window are selectable by mouse.
- *Object Viewer.* The object viewer window provides a grid in which one object, description, or match list is displayed in each grid element. The display of an object or description shows the object's print name, its type, and some of its attribute–value pairs. A view determines which pairs are displayed. The grid element is vertically scrollable. The grid also has an "overflow" column which displays the print names of objects displaced from grid elements by more recently displayed objects. All objects displayed in the object viewer are selectable by mouse.
- *Text Editor.* The text editor is a standard ZMACS text editing window, provided by the Symbolics system software.

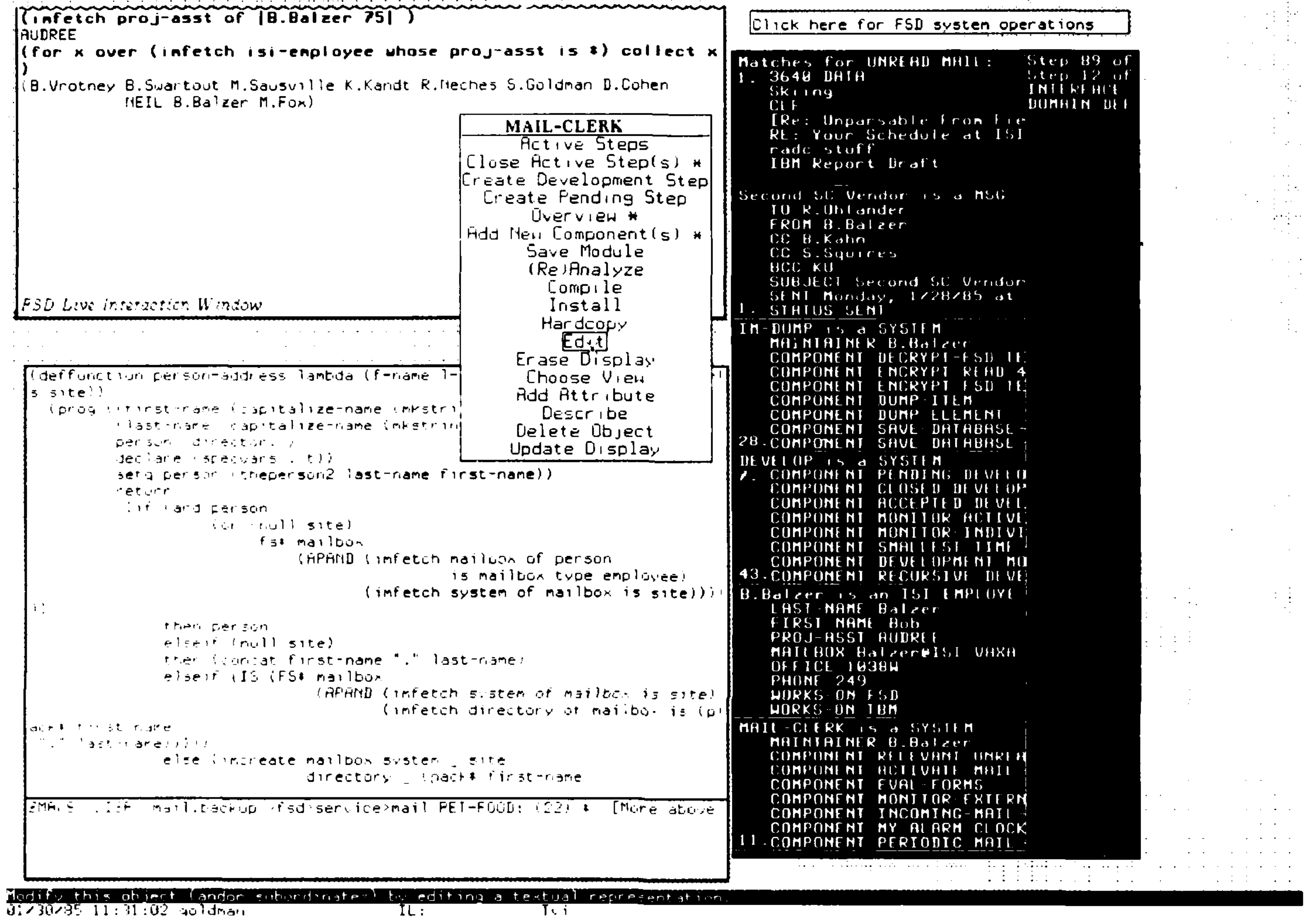
Figure 2-1 shows a typical screen layout.

2.3.4.1 Context-sensitive menus

We developed a capability for controlling the items offered in menus when an IM object is selected with the mouse. The capability permits us to define these menu items for any declared IM type. The items are inherited through the type hierarchy.

2.3.4.2 Text editing linkage

A significant part of the user interface to many applications can be provided by a good text editor. Rather than build a text editor precisely tailored to our needs, we decided to establish protocols that would enable us to use any text editor available on the host machine (in the case of our testbed, this is



```
(imfetch proj-asst of |B.Balzer 75| )
AUDREE
(for x over (imfetch isi-employee whose proj-asst is *) collect x
)
(B.Vrotney B.Swartout M.Sausville K.Kandt R.Neches S.Goldman D.Cohen
          NEIL B.Balzer M.Fox)

FSD Live Interaction Window
```

```
(deffunction person-address lambda (f-name l-
s site))
  (prog ((first-name (capitalize-name (mkstri
        (last-name (capitalize-name (mkstrin
        person directory)
        declare (specvars . t))
        setq person (theperson2 last-name first-name))
        return
         (if (and person
                  (or (null site)
                      fs* mailbox
                         (APAND (imfetch mailbox of person
                                   is mailbox type employee)
                              (imfetch system of mailbox is site)))
))
             then person
             elseif (null site)
             then (concat first-name "." last-name)
             elseif (IS (FS* mailbox
                         (APAND (imfetch system of mailbox is site)
                              (imfetch directory of mailbox is (p
ack* first-name
 "." last-name))))
             else (imcreate mailbox system _ site
                            directory _ (pack* first-name
ZMACS (LISP) mail.backup >fsd>service>mail PET-FOOD: (22) *  [More above
```

```
Matches for UNREAD MAIL:           Step 89 of
1. 3640 DATA                       Step 12 of
   Skiing                          INTERFACE
   CLF                             DOMAIN DEF
   [Re: Unparsable from Fie
   RE: Your Schedule at ISI
   radc stuff
   IBM Report Draft

Second SC Vendor is a MSG
   TO R.Ohlander
   FROM B.Balzer
   CC B.Kahn
   CC S.Squires
   BCC KU
   SUBJECT Second SC Vendor
   SENT Monday, 1/28/85 at
1. STATUS SENT
IM-DUMP is a SYSTEM
   MAINTAINER B.Balzer
   COMPONENT DECRYPT-FSD TE
   COMPONENT ENCRYPT READ 4
   COMPONENT ENCRYPT FSD TE
   COMPONENT DUMP-ITEM
   COMPONENT DUMP ELEMENT
   COMPONENT SAVE DATABASE-
28.COMPONENT SAVE DATABASE
DEVELOP is a SYSTEM
7. COMPONENT PENDING DEVELO
   COMPONENT CLOSED DEVELOP
   COMPONENT ACCEPTED DEVEL
   COMPONENT MONITOR ACTIVE
   COMPONENT MONITOR INDIVI
   COMPONENT SMALLEST TIME
   COMPONENT DEVELOPMENT MO
43.COMPONENT RECURSIVE DEVE
B.Balzer is an ISI EMPLOYE
   LAST-NAME Balzer
   FIRST NAME Bob
   PROJ-ASST AUDREE
   MAILBOX Balzer@ISI VAXA
   OFFICE 1038W
   PHONE 249
   WORKS ON FSD
   WORKS-ON IBM
MAIL-CLERK is a SYSTEM
   MAINTAINER B.Balzer
   COMPONENT RELEVANT UNREA
   COMPONENT ACTIVATE MAIL
   COMPONENT EVAL-FORMS
   COMPONENT MONITOR EXTERN
   COMPONENT INCOMING-MAIL
   COMPONENT MY ALARM CLOCK
11.COMPONENT PERIODIC MAIL
```

```
Modify this object (and/or subordinates) by editing a textual representation.
01/30/85 11:31:02 goldman                  IL:              Tvi
```

Figure 2-1: Typical IM screen layout

the ZMACS editor). We established a protocol for parse and unparse methods for IM objects. The methods are attributes of types and are inherited through the type hierarchy. The unparse method maps an object into a textual representation; the parse method does the opposite.

2.4 IMPACT

A successful conclusion of our research effort will result in more tightly integrated and evolvable systems, produced with fewer man-years of effort. The integration will arise by virtue of portions of specifications being shared by multiple services. The ability to evolve capabilities over time will result from the specificational basis of all capabilities in the environment. Each property will contribute to productivity enhancement—both in what is traditionally thought of as system *construction* and *maintenance*, and in what is thought of as system *use*.

As a side-effect of this primary impact, it should become possible to produce systems with functionality beyond what is achievable from current programming environments. Evolvability will make it economical to provide community-wide or user-specific upgrades to functionality not anticipated by a service designer. Ease of sharing and specification-based software will reduce the cost of providing a given piece of functionality, encouraging service designers to provide additional functionality in their original designs.

2.5 FUTURE WORK

The IM project has been integrated with DARPA-sponsored research on model-based interfaces. The joint effort, now referred to as the Formalized Software Development (FSD) project, is part of a DARPA program to apply artificial intelligence technology to software synthesis and maintenance.

To promote sharing within this community, as well as portability to new hardware, the entire IM system will be translated to the Common LISP dialect. Its programming service will then support the construction of software written in extended Common LISP and transformed into code portable to any Common LISP implementation (possibly requiring some run-time support software, which will also be programmed in Common LISP). Eventually, we want to move towards the use of a specification language like Gist as the language supported by the programming service, still having Common LISP as the target of transformations.

The system will be ported to a relatively inexpensive workstation running Common LISP. It will run in two modes in this workstation. In the first, the entire FSD system will be supported by the workstation. This will allow users access to all FSD capabilities, but with reduced performance (compared to the current Symbolics 3600 hardware base). In the second mode, the workstation will be electronically linked (via a local area network) to a more powerful Common LISP server (Symbolics 3600-class), with the pair cooperating to support higher performance FSD operation. The workstation will support the interactive graphic user interface, while the server supports the rest of the system. A protocol will be designed to enable the pair to exchange objects to support this distributed architecture. This effort assumes the continued reality of dedicated personal workstations with (sequentially) shared servers at a site.

A new interactive interface to the database—the "forms" package—will provide a high-level window system which is portable across workstations having diverse native window systems. A *form* is a high-level specification of the presentation of one or more objects, including text layout, fonts, size, and scrolling characteristics. A form also specifies how the user interacts with the data by defining the interpretation of keyboard and mouse inputs given while the corresponding cursor is positioned within the form.

A new computing environment kernel will give the programmer the ability to annotate the attributes defined in his database model with implementation categories selected from an extensible library. Programs will then be transformed according to these annotations which, if intelligently chosen, can greatly improve the space and time efficiency of the resulting code.

REFERENCES

1. Balzer, R., D. Cohen, M. Feather, N. Goldman, W. Swartout, and D. Wile, "Operational specification as the basis for specification validation," *Theory and Practice of Software Technology* SE-8, (1), 1983, 21-49.

2. Balzer, R., C. Green, and T. Cheatham, "Software technology in the 1990's: Using a new paradigm," *Computer* 16, (11), November 1983, 39–45.

3. Schonberg, E., J. T. Schwartz, and M. Sharir, "An automatic technique for selection of data representations in SETL programs," *ACM Transactions on Programming Languages and Systems* 3, (2), April 1981, 126-143.

4. Weinreb, D., and D. Moon, *Lisp Machine Manual*, 4th edition, Symbolics, Inc., 1981.

5. Wile, D. S., "Program developments: Formal explanations of implementations," *Communications of the ACM* 26, (11), November 1983, 902-911.

3. MAPPINGS

Research Staff:
Robert M. Balzer
David S. Wile
Martin S. Feather
David J. Mostow

Support Staff:
Audree Beal

3.1 PROBLEM BEING SOLVED

Software specification, development, and maintenance continue to present an enormous problem to computer users. We believe that the computer itself must play a far more significant role in the software development process. The software designer's role should be streamlined to require only decision-making and guidance, while the computer's role is expanded to include manipulation, analysis, and documentation. The key to such support is to capture in the machine all crucial information about the processes of specification, design, implementation, and maintenance

We envision a future user developing a high-level specification of what he or she wants a program to do, and transforming the specification into an efficient program, using a catalog of (proven) correctness-preserving transformations. Most debugging and all maintenance will be performed on the specification, which will have an operational interpretation suitable for testing. Reimplementation will be necessary to tune implementations exhibiting unacceptable performance characteristics.

The Mappings project attempts to formally represent all crucial information used in the processes of specification, design, implementation, and maintenance. During the implementation phase of the program development cycle, the high-level constructs used to specify a program must be replaced by efficient realizations of these constructs. The Mappings project has concentrated on discovering and implementing the general transformations—or ***mappings***—required.

We have found that specifications are often difficult to read, despite the high-level nature of the constructs used. The essence of the problem is that the modes of communication normally used between people are considerably richer than those between people and machines. We have several ideas on how this communication gap can be narrowed using incremental specification.

3.2 GOALS AND APPROACH

The primary goal of the Mappings project is to represent programming knowledge sources in terms of mappings between specifications and programs. These mappings fall into two broad categories:

1. mappings for better understanding of the specification, and
2. mappings for implementation of the specification.

We have developed a specification language, called Gist, which formalizes some of the expressive constructs commonly used in natural language specifications. These constructs include nondeterminism, descriptive reference, historical reference, constraints, and demons. A Gist specification describes the behavior of both the program and its environment; this provides a natural way to specify embedded programs that interact in complicated ways with their environment. Gist's expressive power makes it possible to specify ***what*** behavior a program should exhibit without saying ***how*** it should be implemented.

3.2.1 Mappings for Optimization

Since its inception, the Mappings project has addressed the problem of representing programming knowledge in terms of our specification language, Gist, by discovering correctness-preserving transformations for implementing Gist's high-level constructs in terms of the lower level constructs used in more conventional programming languages. For each construct, our research aims to accumulate: ***implementation options*** for converting an instance of the specification construct into a more efficient expression of the same behavior in terms of lower level constructs; ***selection criteria*** for deciding among these options; and, ***mappings*** to achieve the implementation options via sequences of correctness-preserving transformations. Presently, more emphasis is being given to mappings for optimization of these implementations.

3.2.2 Mappings for Understanding

Understanding of a specification should ***evolve*** rather than occur all at once. The power of prose descriptions derives, in part, from reliance on a reader's ***incremental*** understanding as he goes through a document sentence by sentence, subsection by subsection. We believe we can convey such incremental understanding of specifications in terms of ***high-level editing*** commands that formally describe the evolution of Gist specifications:

- ***refinement*** of modeling detail,
- ***generalization*** of functionality,
- ***introduction of exceptions*** into idealized specifications,
- ***restriction*** of scope or functionality.

We have begun to capture these methods as ***formal*** methods, thus permitting the specification development to appear much more natural to future readers of the specification. Unlike the mappings for optimization, these mappings for understanding often ***change*** one specification into one with observably different behavior.

A second means for enhancing understandability is to allow much more ***expressive*** modes of specification than are presently used. Generally, the formal languages used to describe program behavior to machines (i.e., programming and specification languages) are extremely restrictive and primitive. For example, most journal articles contain tables, grammars, equations, graphs, and other specialized notations to communicate how a program works or what it does. There is no apparent reason why computers cannot begin to understand these richer, higher level modes of expression. We have coined the phrase ***locally formal notations*** to describe the specification technique we espouse. We emphasize that, although the notation is self-consistent, it can only be interpreted in context. That context establishes a meaning and allows the notation to be concise. Of course, these notations must be internally mapped onto conventional Gist semantics.

3.3 SCIENTIFIC PROGRESS

Considerable progress has been made in the development of high-level transformations:

- **Transformations to remove Gist high-level constructs,** such as nondeterminism. recursively derived relations, and references to historical information.
- **Transformations for rapid prototyping,** choosing adequate implementations for Gist high-level constructs to ***validate*** that a specification's behavior is what was intended.
- **Transformations to remove Gist's "perfect knowledge" assumption,** by implementing complex, quantified relations in more conventional data structures.

During this reporting period, we have begun to develop what we call **paradigmatic transformations**: transformations which capture common high-level optimization techniques. Our first example of such a transformation is "memoize," a transformation which introduces a data structure for caching information and causes subsequent retrieval of cached values for previously computed arguments, rather than recomputation. The development of this transformation is particularly interesting in that the concerns which have gone into producing the transformation seem to typify the thought processes of implementers optimizing functions.

3.3.1 The Memoize Transformation

Basically, the memoize transformation substitutes a fast lookup of the value of a function (from a cache) for the recomputation of it. In essence, if we have a function

```
f(x) = <...x...>
```

we can substitute

```
f(x) = if x is in f's cache
       then x's value in f's cache
       else put <...x...> in f's cache as x's value and return it
```

Of course, this can only occur in some context in which the cache remains valid, such as:

```
begin @local f-cache:
  ... uses of f ...
end
```

Three major problems must be solved:

1. When is it "locally" worth caching a function? In particular, when does the cost of the access time make up for the extra cost of executing the function? Obviously, computing the square root function would normally not be worth caching. However, if the precision were several hundred digits...
2. When is the scope of the cache large enough to make the extra mechanism for storage maintenance worthwhile?
    - Is the result likely to be needed several times in the same context?
    - In particular, how frequently are results invalidated because of global dependencies of f on its context or destructive modification of dependent data structures?
3. How do multiply cached functions interact, i.e., what is an appropriate overall caching strategy?

The first problem is fairly easily solved for functions which depend only on primitive functions (which will not be cached), as long as one is willing to settle for normally distributed access assumptions within the function. When more than one potentially cached function is involved, we have no analagous analytic solution. Instead, we rely on the ***simultaneous*** analysis of all functions within a given context for potential memoizing, and ***heuristics*** which consider potential interactions. This same simultaneous analysis is used to solve the second problem as well, for as the analysis decends (recursively) through a candidate function looking for subexpressions to cache, different cache scopes can be proposed and analyzed.

Some example heuristics include:

- An expression is potentially worth caching if it costs more to compute than to retrieve.
- Other functions which contain neither loops nor recursive calls should not be memoized.
- A function executed for side–effect should not be memoized.[1]
- If any of an expression's side–effects disqualify it, see if it has memoizable subexpressions.
- If a data structure returned by an expression is destroyed (by pointer manipulation, aliasing, etc.), the expression should not be memoized.
- If a memoized expression contains subexpressions with fewer parameters, consider memoizing them as well.

Presently, weak analysis tools force the memoize process to be either interactive or too conservative. Knowledge about the potential for copying results that are destroyed by pointer manipulation is essential for this package to work well, and that cannot be deduced from LISP source code with any confidence. Frequency statistics are also essential for good performance by the package; hence, we now instrument the running code to corroborate our predictions. The memoize package has been used to memoize itself and is certain to gain significant usage in the Formalized Software Development (FSD) testbed system in the near future.

3.3.2 Implementation Semantics and Responsibilities

Over the past several years we have gained considerable experience with the design of medium–sized specifications in Gist. The Mappings project has played a major role in the design of Gist: in order to understand the validity of transformations the Mappings project has formalized Gist's semantics. This has led to greater understanding of the specification process and its relationship with both the domain being modeled and the intended implementation environment.

3.3.3 Implementation Specification

Gist specifications are unusual in that the behavior of an entire system and its environment is specified: every independent agent able to affect the behavior of the system must be modeled to the extent that he can affect it. This means that every person and every piece of hardware in the imagined system must be modeled. It is then the task of the implementor to implement a portion of the specification as a computer program.

We have developed a formal definition of the meaning of "implementation" of a Gist specification in

[1]A relaxed version of this would change the model above slightly, to retrieve from the cache when cached and to call the side–effect functions

which a "specification of the implementation" must be provided separate from the Gist specification of the functionality of the system. Such a specification partitions the world into the "system to be implemented" and the "environment in which it will be implemented." This partition also describes the interface between the two portions: i.e., the limits on one portion regarding control or information availability in the other portion. Any implementation which obeys the interface limitations is said to be ***compliant***.

A ***correct implementation*** imposes different requirements on these two domains. The environment is generally autonomous: the system must react gracefully to *all* nondeterministic activity from the environment. On the other hand, the nondeterminism of the system portion of the specification can be exploited for efficient implementation: *any* behavior from the nondeterministically specified set is acceptable.

We have dealt with the problematical issue of establishing the appropriate set of interface limitations on the implementation. The basic issue is transforming a global database and globally visible process activity into local databases of processes with appropriate communication protocols. The key to the solution of this problem has been the development of the notion of ***responsibility*** of agents for the maintenance of constraints in the implementation. The basic idea is simple: if an agent is responsible for the maintenance of a constraint, it must be impossible for other agents to manage to produce a situation in which the constraint is violated. Some behaviors cannot acceptably be partitioned into the responsibility of single agents. In these cases, implementation consists of substituting protocols and subsequent responsibilities of the interacting agents.

3.3.4 Mappings for Understanding

3.3.4.1 Specification evolution

One reason specifications are hard to read is that the ***evolution*** of the specification is often the only way to understand what is happening. In particular, numerous implementation decisions enter into the actual functionality specified for programs [1]. For example, any portion of a specification that is present to report error conditions to the user, or take actions based on error conditions, could not possibly be a part of the specification of the *ideal* behavior of the system (why would an ideal system make errors?). Hence, at some point in the evolution of the system, someone realized that the particular implementation that would be used would necessitate dealing with such imperfect behavior. Of course, systems are never ideal: finite space and time resources force less-than-perfect implementations (e.g., finite stacks, heuristic game-playing programs, and out-of-paper status flags). We believe that the best way to explain such a specification is as an ***incremental*** modification to the original ideal specification.

In the past, we identified several categories of high-level editing steps to describe how specifications are actually created, such as refinement, generalization, specialization, exception introduction, implementation decision, and redundancy introduction. We experimented with these categories as ***informal*** indications of the development of actual programs using the Develop System [2]. The system records design decisions made in the course of developing a program: it encourages the developer to classify each design decision into one of these categories and document it in English. We began to formalize these activities as high-level editing commands for modifying the FSD testbed model.

3.4 IMPACT

The Mappings project is supportive of the transformational implementation software development paradigm [3]. Initially, this paradigm will dramatically improve programmers' ability to maintain software systems. Ultimately, the entire lifecycle from design through specification, implementation, debugging, and maintenance will be streamlined. This will allow programmers to produce and maintain software systems more rapidly and will make feasible the construction of larger, more complex systems with enhanced functionality and flexibility. Of course, a major goal of the Mappings project is to facilitate system maintenance by providing system designers with modes of expression that make specifications more easily understood and hence more modifiable. In addition, this project will codify the knowledge needed by programmers to convert high-level specification concepts into efficient implementations via mappings. This knowledge would also be useful for programmer education independent of its mechanized application via transformations.

3.5 FUTURE WORK

Our future plans center on our interest in producing mappings which are useful in the everyday development of running the software we use in our own environment. We have begun to implement a language called Will, an automatically compilable subset of Gist. (This subset was formed by omitting Gist's highest level constructs.) The compilation process is directed by user-supplied annotations to the program describing what implementations to use for specific data structures. We intend to use this language as the target language for our Gist mappings and as our high-level programming language for system development. We want to begin developing the system within itself, using the transformational programming paradigm. Although we have laid the groundwork for some of the high-level mappings from Gist *into* Will, we have found that the lack of low-level transformations presents an enormous bottleneck to the development of the system *within* itself. Hence, our attention is currently focused on some quite utilitarian goals to provide the Gist- and Will-specific mappings and mapping technology needed to produce a usable system in the near term.

The understanding conveyed by the high-level editing commands and locally formal notations—to both human readers and automatic analysis tools—will form a basis for ***evolving and adapting*** software. In fact, such ***high-level mappings for understanding*** have begun to dominate the research goals of the Mappings project. Several practical high-level editing operations have been defined for the FSD testbed system and are in daily use for changing the domain model in the specification. It is fair to characterize our efforts in this direction as "experience-based." Those constructs needed in our everyday development of the system are emphasized. As our understanding of the broad categories of these practical applications grows, we expect a theory of such modifications to emerge.

A longer term goal is to develop a framework for locally formal notations, in which context information selects a semantic backdrop to facilitate concise expression of the important aspects of the specification.

REFERENCES

1. Balzer, R., and W. Swartout. "On the inevitable intertwining of specification and implementation." *Communications of the ACM* 25, (7), July 1982.

2. Balzer, R., The Develop system. 1983. In progress.

3. Balzer, R., C. Green, and T. Cheatham. "Software technology in the 1990's: Using a new paradigm," *Computer* 16, (11), November 1983, 39–45.

4. TRANSFORMATION-BASED MAINTENANCE

Research Staff:	*Graduate Student*:	*Support Staff*:
Robert M. Balzer	Randell Kerber	Audree Beal
David S. Wile		
Martin S. Feather		
David J. Mostow		

4.1 PROBLEM BEING SOLVED

Software specification, development, and maintenance continue to present an enormous problem to computer users. We believe that the computer itself must play a far more significant role in the software development process. The software designer's role should be streamlined to require only decision-making and guidance, while the computer's role is expanded to include manipulation, analysis, and documentation. The key to such support is to capture in the machine all crucial information about the processes of specification, design, implementation, and maintenance.

For several years we have been developing an alternative software development paradigm based on this tenet [1].[1] We envision a future user developing a high-level specification of what he or she wants a program to do, and transforming the specification into an efficient program, using a catalog of (proven) correctness-preserving transformations (see Figure 4-1). Most debugging and all maintenance will be performed on the specification, which will have an operational interpretation suitable for testing. Reimplementation will be necessary to tune implementations exhibiting unacceptable performance characteristics.

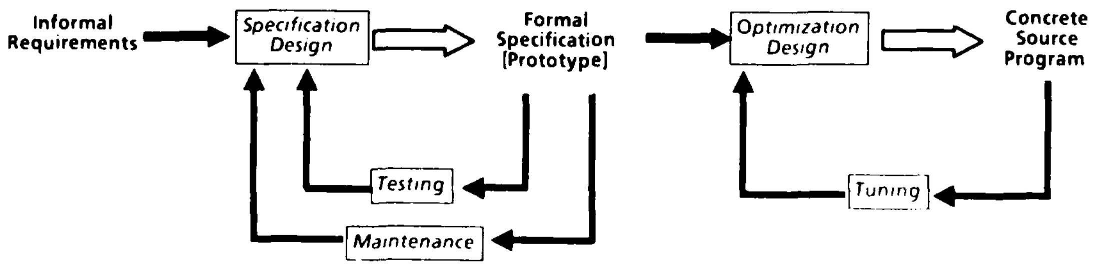


Figure 4-1: Alternative software development paradigm

[1]Several researchers led by Robert M. Balzer, working on projects funded by DARPA, NSF, and RADC, have provided the foundations for this paradigm.

This automation-based paradigm will rely on an integrated set of tools that directly support human developers in the processes of requirements specification, system design, implementation, and maintenance. It will be characterized by the fact that the entire evolution of a system—the history of all four of these processes, as directed by the developers—will occur and be recorded within the integrated environment. This history will provide the "corporate memory" of each system's evolution. The software development system will be an active participant in the development process, using the history to determine how these four processes interact: what assumptions they make about each other, what the rationale behind each evolution step was, how the developed system satisfies its requirements, and how to explain all of these to its developers.

We have developed a specification language called Gist, which formalizes some of the expressive constructs commonly used in natural language specifications, including nondeterminism ("a message from another site"), descriptive reference ("the longest message"), historical reference ("the last message the user looked at"), constraints ("never send multiple copies of a message to the same person"), and demons ("if a week passes without a reply to a request, inform the sender"). A Gist specification describes the behavior of both the program and its environment; this provides a natural way to specify embedded programs that interact in complicated ways with their environment. Gist's expressive power makes it possible to specify ***what*** behavior a program should exhibit without saying ***how*** it should be implemented [2].

Given that we already have a language for expressing specifications, three major problematical areas require research and development before our transformation-based paradigm can succeed:

1. Tools, methods, and support facilities for ***acquiring***, ***designing***, and ***developing*** the specification consistent with a user's intent;
2. Tools, methods, and support facilities for ***implementing*** and ***optimizing*** the specification;
3. A framework for ***understanding***, ***reusing***, and ***maintaining*** previous specifications and optimizations.

The primary goal of the Supervisory Control of Transformational Implementation Systems (SCTI) project was to design and develop a framework for understanding, reusing, and maintaining previous specifications and optimizations. The SCTI project was a predecessor to the Transformation-Based Maintenance (TBM) project, which developed the system support needed to facilitate the automated implementation of specifications using transformations. The TBM project is extending the framework to deal with the activity of ***redesign*** of specification and implementation.

4.2 GOALS AND APPROACH

4.2.1 Overview

Over the years, the group has developed several tools to support the specification ***process***, i.e., to help bridge the gap between the informal intent inside the specifier's head and its formal expression in Gist. Specifications in any formal language tend to be difficult to understand, because they are usually concise and lack the redundancy found in natural language descriptions of the same behavior. Hence, we have designed a program that exposes static aspects of Gist specifications by paraphrasing them into English [4]. To expose dynamic implications of a specification, we have developed a symbolic evaluator that looks for interesting consequences of a Gist specification [3], and a program that explains the symbolic trace in English [5]. By clarifying what a specification really means, such tools help ***validate*** the specification, i.e., increase the specifiers' confidence that the specification matches their informal intent and the users' confidence that the resulting system will meet their needs.

However, we have discovered that when people specify programs to other people, they use an ***incremental*** technique for conveying the information necessary to develop the program. This technique is built on a very stylized set of methods, such as ***refinement*** of modeling detail, ***generalization*** of functionality, ***introduction of exception*** into idealized specification, and ***restriction*** of scope or functionality. In the future, our Mappings project will address the problem of capturing these methods as ***formal*** methods, thus permitting the specification development to appear much more natural to future readers of the specification. We have called these methods ***high-level editing*** commands, to emphasize the fact that they do indeed change the specification, yet at the same time convey more information to a reader than conventional keystroke editing commands. The use of high-level editing commands to design specifications is analogous to the use of transformations to develop implementations (see Figure 4-2).

Since its inception the Mappings project has addressed the problem of representing programming knowledge in terms of our specification language, Gist, by discovering correctness-preserving transformations for translating Gist's high-level constructs into the lower level constructs used in more conventional programming languages. Hence, the Mappings project addresses the second problem mentioned above: encapsulating programming knowledge in terms of concise mappings from Gist into alternative ***implementations***. In the future, more emphasis will be given to mappings for ***optimization*** of these implementations.

Our paradigm is based on capturing the ***processes*** of specification, design, implementation, and maintenance within the computer, and supporting them via automated tools. We have developed an experimental system in which the user develops not only the implementation, but also a formal explanation of how it arose: the design decisions, assumptions, and optimization steps, and their structural relationships, are recorded in a formal design document suitable for automatic reuse by the transformation system (see Figure 4-3). Hence, using this ***formal development*** the system can replay optimizations on a changed specification to produce a new implementation automatically [6]. Naturally, all future mappings and transformations will be designed to fit into this framework.

Our primary goal is to obtain a usable development system. We believe that the leverage provided by our alternative software development paradigm (especially in the beginning) will arise from its support of the maintenance process rather than from its support of the initial implementation. This is because the increased formalization of the development process requires more involvement by the human developer (at least until the level of automation is significantly increased) for the initial implementation. On the other hand, maintenance involves small changes. Increased formalization of these changes is insignificant if the already formalized development can be adapted and reused.

The SCTI project laid the groundwork toward this goal, including the creation of a language-independent development system called Popart, with facilities for grammar-based editing, analysis, and transformation; a formal representation of program development (explanations of how the implementation was produced); a compilable subset of our (wide-spectrum) formal specification language to act as a target language for transformation; a goal-directed transformation selection and application mechanism; and a system for recording and supporting the evolution of real systems (to improve our understanding of this evolutionary process). However, much remains to be done. Current limitations include inadequate automation of transformation selection and application,

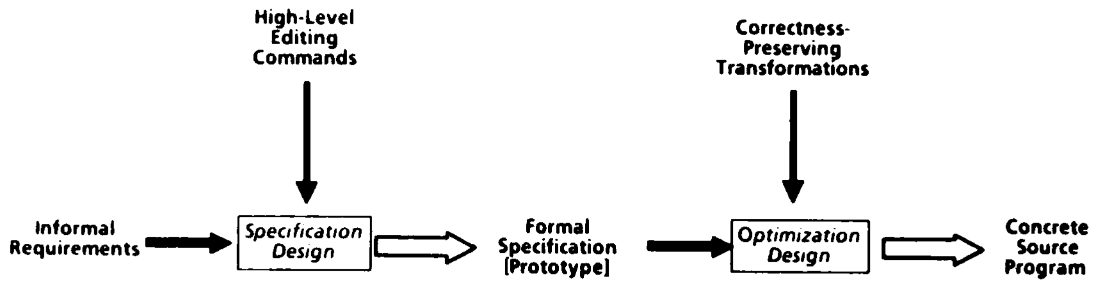

High-Level Editing Commands
Correctness-Preserving Transformations
Informal Requirements
Specification Design
Formal Specification [Prototype]
Optimization Design
Concrete Source Program

Figure 4-2: Design process mappings

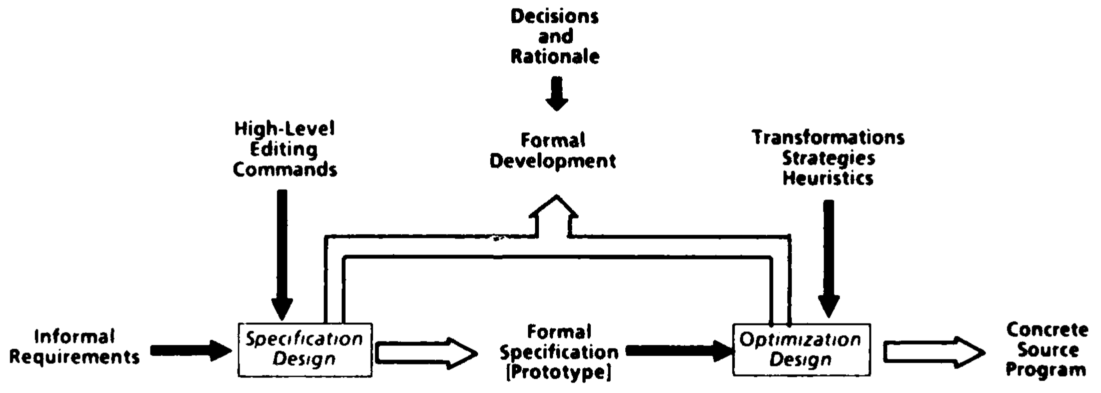


Figure 4-3: Formal developments

inadequate analysis tools (verifying the precondition of transformations), an incomplete transformation catalog, no performance prediction tools, no direct support of maintenance, and inadequate integration of current capabilities into a coherent system.

We have begun research on three forms of reimplementation support: strengthening the formal representation of developments (especially the rationale behind each step), supporting the evolution of specifications through the same mechanisms that support maintenance (and in fact integrating the two processes), and developing a formal language of modification for both specifications and developments.

4.3 SCIENTIFIC PROGRESS

Much of the TBM work relies on the foundations of the predecessor SCTI project. Hence, a brief summary of some of its accomplishments is necessary to establish context for the TBM work.

4.3.1 SCTI Foundations

The SCTI project focused on the formalization, mechanization, and automated support of the development process through transformations. Under that project we developed our primary transformational tool, Popart, a prototype program development system which applies transformations selected by the user. The detailed strategy used for such transformation applications, called a development, is formalized in a modifiable and executable language, called Paddle. This language represents a development as a hierarchical plan that incorporates subplans, strategies, refinements, and transformations. This development defines, in an executable form, the sequence of transformations and manipulations that must be performed to convert the specification into a program. In this mode, each refinement defines how to accomplish the refined goal by modifying the specification/program resulting from previous steps. The execution of such a development can be faulty because some development step either has not yet been defined or is inappropriate for the current state of the specification/program being developed. When this occurs, the development can be edited using the Popart system (described below) and its execution can be either continued or restarted from some previous point.

The development is thus a "program of transformations" for implementing a specification which can be debugged via standard interactive debugging techniques. It is a structured, documented, and (when it successfully runs without manual intervention) fully automatic method for implementing the specification. This formal manipulable representation of the development was a primary scientific achievement of the SCTI project and is more fully documented in [6].

This transformation activity leads to the conversion of specifications written in Gist into programs written in a compilable subset called Will. This subset shares much syntactic structure with Gist, reducing the need for "form-changing" transformations, but it replaces many of Gist's semantic freedoms with similar, but lower level, constructs which can be more efficiently compiled. Generally, Will supports the data typing, structuring, and associative retrieval of Gist and its event-driven (demon-based) processing. It does not support the full semantics of historical references, constraints, or inferences. Instead, it supports more restricted versions, such as constraints that cause an exception to be raised when violated (in Gist, constraints affect the nondeterministic control structure, so that execution proceeds only along paths that do not, and will not, violate them), and monotonic inferences (once true, they remain true even if their support subsequently vanishes).

Will's significance lies less in its novel features than in its expected role in our efforts to develop a usable transformational implementation system. We now have a (high-level) implementation language to use as a target for our developments that will be syntactically compatible with our specification language. This compatability will permit us to focus our transformation-building efforts on the few (but important) semantic distinctions between the two languages.

Finally, we built a separate system (not based on transformations), called the Develop System, to record the actual evolution of real systems in order to sharpen our perceptions about how that activity should be supported by our tools. It automatically captures the development and maintenance of actual programs written in Interlisp, in terms of the objects (data structures, functions, macros, etc.)

created and modified (including the modifications themselves). This trace of development behavior is annotated by statements of intent by the user, indicating the type of development step (augmentation, generalization, revision, exception, specialization, establish-convention, reorganization, bug-fix, tuning, etc.) being initiated and a descriptive comment.

During the modifications, the user can recursively initiate substeps that modularize his efforts to achieve the intent of the higher level step (goal). Also, the user can post "pending steps" on an electronic blackboard for future implementation (this useful notepad allows us to record future plans as they arise). It is important to recognize that this represents a different paradigm for construction of the formal development than that supported by Popart and Paddle.

Our primary reason for creating this system was to gather histories of the actual evolution of real systems in order to provide insights into how users conceptually structure developments and into the dynamics of system growth and modification. These insights will be fed back into the redesign of our formal representation of developments and the support mechanisms of Popart. In fact, they have already become the basis for the "high-level editing" language outlined in Section 4.2.1. However, we unexpectedly found that these development histories could also provide support for novel maintenance tools within the Develop System.

4.3.2 Transformation-Based Maintenance Progress

The TBM project has concentrated on preparing for and obtaining actual use of the prototype transformational tools we had previously developed. The purpose of such real usage is both to obtain feedback, which helps us better understand the capabilities and limitations of our tools and the transformational paradigm, and to leverage our own software development using these advanced tools.

We also have adopted this same strategy with our other software technology projects, and have built a software development testbed—Formalized Software Development (FSD)—incorporating those aspects of our work that were ready to be practically exploited or realistically tested. This testbed is in daily use within our group and we plan to export it this year to other research organizations.

4.3.2.1 Incorporating facilities into our testbed environment

Naturally, we want to incorporate relevant portions of the TBM project into the FSD testbed. Three capabilities were selected: the Develop system, Popart, and the Will language. In this reporting period, the Develop system was fully incorporated into the FSD testbed system. The stand-alone version (which was used only by the developer) has been integrated into the computing environment and is now in use by others within the group. This integration has drastically altered the Develop system. It is one-fourth its former size and is 75 percent declarative rather than completely procedural. Furthermore, its use of the computing environment's general object manipulation facilities has made it both more powerful and more comprehensible.

One particularly important extension to Develop is the ability for a developer to *accept* a completed development step. This causes all copies of the system to be updated with those changes (i.e., this is the mechanism for maintaining released copies of the system).

This new Develop service, together with Paddle, is the foundation upon which our work on integrating the development of specifications and implementations, the formalization of a modification language, and the replay of developments will occur.

4.3.2.2 An experiment for replay

Another important result is the development of a Gist specification and implementation for the behavior of robot "arms." This real problem arose from frustrations of Professor Soroka at the USC Robotics Institute in coding robot control programs in typical Pascal-level programs. As he described the activity in English that the robot arms were to exhibit, it became obvious the the Gist specification technique was well matched to the problem. In particular, one arm is to pick up a block and pass it on to the other arm, which places the block on a table. This straightforward behavior specification is easily described nondeterministically in Gist. This behavior is restricted to the desired subset via Gist constraints such as "a block must always be supported" or "arms must not hit one another."

The resulting specification was then transformed (by hand) into an implementation that was quite close to the implementation in Pascal (obviously, with foresight). Of particular relevance to our goals is a formalized development of the specification that was produced, in which all the design goals and their relationships were carefully recorded. We now want to change the specification to involve three robot arms passing blocks in a circle. This should provide an excellent example for understanding how much of the original development can be replayed.

4.4 IMPACT

The TBM project is supportive of the transformational implementation software development paradigm [1]. Initially, this paradigm will dramatically improve programmers' ability to maintain software systems. Ultimately, the entire lifecycle from design through specification, implementation, debugging, and maintenance will be streamlined. This will allow programmers to more rapidly produce and maintain software systems and will make feasible the construction of larger, more complex systems with enhanced functionality and flexibility. In addition, we will codify the strategic, language-independent knowledge used by programmers in specification and optimization. This knowledge will also be useful for programmer education, independent of its mechanized application.

4.5 FUTURE WORK

We intend to continue to experiment with the prototype transformational tools we previously developed. The purpose of such real usage is to obtain feedback, which helps us better understand the capabilities and limitations of our tools and the transformational paradigm, and to leverage our own software development using these advanced tools.

4.5.1 Incorporating Facilities into Our Testbed Environment

As mentioned above, one of our technical goals is to incorporate the Develop System, Popart, and the Will language into the testbed environment. Although the Develop System is fully integrated, we want to enhance it to also record changes in the makeup of the system (i.e., the set of components that comprise the system); allow multiple developments to be open simultaneously; automatically determine in which development a change should be recorded; allow the developer to revert to earlier states; and automatically install changes upon completion of top-level development steps. Furthermore, we would like to use FSD's consistency maintenance facilities to simplify the implementation of Develop and to improve its robustness.

We expect the introduction of Popart (and Paddle) into the FSD environment to be fairly straightforward. However, in order to integrate Popart with the environment, and Paddle with Develop, several problems must be solved:

- Popart provides a "glass teletype" structured editor. This is far too cumbersome in comparison with the two-dimensional world provided by our new computing environment. While screen-oriented editors tug structure editor design towards mouse-clicks and text motion, the need to formally describe modifications to programs in a language for program transformation pulls toward verbosity. The only solution is to make structure-oriented editing commands so powerful that they are frequently the obvious way to change a program, and to allow these commands to be invoked from the screen-oriented editor.
- In order to use the same structural representation for our formal developments in both Paddle and Develop, FSD's object/attribute description of structures must be unified with the abstract syntactic grammatical structures used by Paddle. We have ideas on how to accomplish this unification, in a concept we call "worlds," and we expect the marriage between the goal-structure components of the Develop and Paddle languages to happen soon.

The Will language has strongly influenced the design of the FSD system. Much of the abstract semantic model underlying the system is the same as Will: a type hierarchy which classifies objects manipulated by programs, generic actions applied to elements of types, demonic reaction to modifications to a global database of objects, and a strong notion of attributes relating objects.

Along with the incorporation of Popart into FSD, we intend to incorporate a type checker and compiler for Will (in the Mappings project). This will pave the way toward realizing one of our major goals: to make Will the syntactic front end to the FSD system. This will allow us to prototype functions in Will. Ultimately, this will allow us to transform Gist specifications into a real, implemented language.

4.5.2 Experimental Transformation-Based Maintenance

As mentioned above, we have developed a specification of the movement of two robot arms (to accomplish a simple box movement task) and sketched a transformational development to implement this specification in Pascal. We kept track of the goal structure of the optimizer (person) during this development, and intend to formalize this specification and development, and then to attempt to change the specification to one with three cooperating robot arms, and to replay the original development on this modified specification.

REFERENCES

1. Balzer, R., C. Green, and T. Cheatham, "Software technology in the 1990's: Using a new paradigm," *Computer* 16, (11), November 1983, 39–45.

2. Balzer, R., N. Goldman, and D. Wile, "Operational specification as the basis for rapid prototyping," in *Proceedings, ACM SIGSOFT Software Engineering Symposium on Rapid Prototyping*, 1982.

3. Cohen, Donald, "Symbolic execution of the Gist specification language," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 17–20, Karlsruhe, Germany, August 1983.

4. Swartout, W. R., "Gist English generator," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 404–407, AAAI, August 1982.

5. Swartout, W. R., "The GIST behavior explainer," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Washington, D.C., August 1983. Also available as USC/Information Sciences Institute, RS–83–3, July 1983.

6. Wile, D. S., "Program developments: Formal explanations of implementations," *Communications of the ACM* 26, (11), November 1983. Also available as USC/Information Sciences Institute, RR–82–99, August 1982.

SYSTEMS DIVISION

The projects comprising the Systems Division cover a wide range of research areas with a common objective: the research and design of working. fully realized systems.

The VLSI and Advanced VLSI projects designed. implemented. and continue to operate MOSIS—the DARPA silicon broker that provides integrated circuit and printed circuit board fabrication to the VLSI design community. MOSIS's main function is to act as a single interface between a geographically distributed design community and a diverse semiconductor industry. As such an interface. MOSIS significantly reduces the cost and time associated with prototyping custom integrated circuits and printed circuit boards. and serves as a central resource for sharing experience. cells, design tools. and software.

The Internetwork Concepts Research project extends and enhances computer communications based on the ARPA-Internet, pursuing in parallel the abstract design and the implementation of protocols. The interaction of these activities provides valuable insights into problems and potential solutions. The current research focus is on abstract process communication. hierarchical naming. multimedia mail. and high-speed network interfaces. The Internet Concepts Research project is also concerned with new capabilities in the gateway protocols, mechanisms and procedures for monitoring and maintaining performace and survivability. and the continued growth and dynamic control of the ARPA-Internet.

The Wideband Communication project participates in the DARPA/DCA Wideband Packet Satellite Program; the WBC project performs the role of the experimenter, applying the Wideband Packet Satellite Network to new modes of communication, such as packet voice and packet video.

The Command Graphics project develops graphics systems for military environments where command mobility is paramount and a portable. network-based graphics capability is necessary for information presentation to facilitate command decisions.

The Strategic Computing Information System will provide for the automatic dissemination of Strategic Computing program-related information to its research community, much as MOSIS does for its users.

The NeaT project is studying the problem of information transfer from sophisticated aircraft systems to pilots. The project will design an advanced, information-oriented, man-machine interface for aircraft, which will behave as an intelligent associate to the pilot. The interface will be based on an expert system, which models the decisions of an experienced copilot. A major focus of the project will be the interface between this expert system and the pilot: the project will study various methods and philosophies of information presentation. with an emphasis on intelligent speech and modifiable visual displays.

The following chapters discuss the accomplishments of seemingly unrelated research projects, yet each attests to the benefits of well-focused research in creating systems for specific applications and environments.

5. VLSI

***Research Staff*:**	***Support Staff*:**
George Lewicki	Kathie Fry
Ron Ayres	Terri Lewis
Danny Cohen	Jasmin Witthoft
Joel Goldberg	
Lee Richardson	
Barden Smith	
Victoria Svoboda	
Vance Tyree	

5.1 PROBLEM BEING SOLVED

The VLSI design communities of DARPA and NSF require fabrication capabilities in order to investigate the design methodologies and architectures appropriate to VLSI where gate-count will exceed one million gates per device. Recognizing this need. DARPA established the MOSIS (MOS Implementation Service) system at ISI in January 1981. MOSIS has

- reduced the cost of VLSI prototyping;
- shortened turnaround time for VLSI prototyping;
- freed designers from fabrication idiosyncrasies: and
- made design less dependent on specific fabrication lines.

A cost reduction of one to two orders of magnitude has been achieved by spreading the fabrication cost over many projects. By centralizing (and computerizing) the idiosyncratic knowledge about all vendors, MOSIS eliminates the need for designers to familiarize themselves with many details. Serving as the only interface between its design community and the vendor base, MOSIS is able to provide turnaround times of four to six weeks for standard technology runs, except when unusual fabrication problems occur. Nonstandard technologies and experimental runs generally require longer fabrication schedules.

5.2 GOALS AND APPROACH

MOSIS involves various aspects of multiproject wafer assembly, quality control, and interaction with industry. The major components of the MOSIS system are

- interaction with the designers;
- handling of their design (CIF) files:
- communication over either the ARPANET or TeleMail:
- placement of projects on dies. and dies on wafers:
- matching of MOSIS design rules to specific vendors' design rules. addition of alignment marks. critical dimensions, and test devices;
- fabrication of E-beam mask sets (via subcontract):
- fabrication of wafer lots (via subcontract);
- wafer probing and data analysis;
- generation of bonding maps;

- wafer sawing, die packaging, and bonding (via subcontract);
- device distribution.

Designers use any available design tools to create artwork files, which are sent to MOSIS via the ARPANET or other computer networks. MOSIS compiles a multiproject wafer and contracts with the semiconductor industry for mask making, wafer fabrication, and packaging. MOSIS then delivers packaged IC devices to the user. The user perceives MOSIS as a black box that accepts artwork files electronically and responds with packaged IC devices.

Though MOSIS may be instrumental in providing cells and design tools to the user, it is the sole responsibility of the user to see that the submitted patterns yield working designs. One may compare MOSIS to a publisher of conference proceedings compiled from papers submitted in "camera-ready" form, where the publisher's responsibility is to produce the exact image on the right kind of paper using the appropriate ink and binding—but not to address the spelling, grammar, syntax, ideas, or concepts of the various papers.

MOSIS provides a clean separation of responsibility for the "printing" of chips. The semiconductor manufacturer is responsible for the processing of the parts and must satisfy MOSIS's rigorous quality control. MOSIS is responsible to the user for the quality and timeliness of the fabrication. The user is responsible for the proper design of the parts and may use any design methods he finds appropriate for his needs.

It is quite common that very advanced and sophisticated chips fabricated by MOSIS work on "first-silicon." An example of this is Caltech's MOSAIC—this is an amazing accomplishment of the existing design tools. Unfortunately, this is done at a considerable cost; for example, it is estimated that Caltech's MOSAIC chip consumed over 1,000 CPU hours on various VAXes before it was submitted to MOSIS for fabrication.

5.3 SCIENTIFIC PROGRESS

5.3.1 Technology Base for Fabrication Runs

nMOS

MOSIS routinely supports nMOS at 3.0 and 4.0 micron feature sizes, with *buried*, rather than *butting*, contacts, in accordance with the Mead-Conway design rules. At least 20 vendors can fabricate devices according to these nMOS design rules.

CMOS/Bulk

MOSIS supports CMOS/Bulk (typically P-well) with 3.0 micron feature size, usually with a capacitor layer and occasionally with a second metal layer.

CMOS/SOS

MOSIS supports CMOS/SOS fabrication with 4.0 micron feature size in accordance with the Caltech design rules. All of the SOS vendors support these design rules.

Completed Fabrication Runs

The following is a categoric breakdown by technology of the fabrication runs completed during this reporting period:

17 runs	nMOS, 4 microns
5 runs	nMOS, 3 microns
11 runs	CMOS/Bulk, 3 microns
2 runs	CMOS/SOS

Figure 5-1 depicts a typical nMOS run: Part 1 is a 4-inch wafer; Part 2 is a multi-project chip on the wafer with one project bonded; and Part 3 is a packaged device which the designer receives. Figure 5-2 is a wafer from a CMOS run, M38C-Clyde.

5.3.2 Computing Interface

Since its inception, MOSIS has used KL-2060s (running TOPS-20) not only to communicate with users on the network, but also for its considerable computing requirements. It has been determined that VAX 11/750s would provide a more efficient and economical environment for computing. Substantial progress has been made in porting the MOSIS geometry software to the VAX 11/750.

MOSIS now generates MEBES files on the VAX 11/750. Computing on VAX 11/750s has several benefits over KL-2060s:

- maintainability
- extensibility
- economy
- single usage eliminates the burdens of shared resources
- software implementation is more exposed.

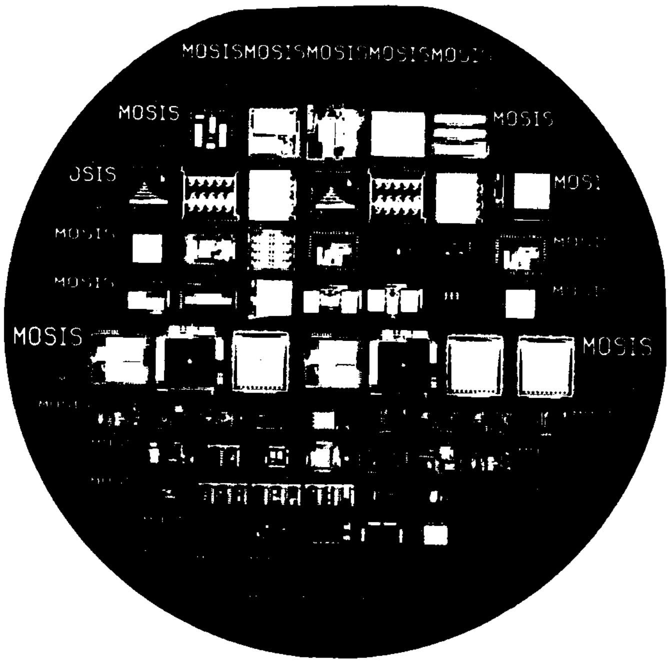
The transition to VAX 11/750s has been be transparent to MOSIS users: they continue to access MOSIS on KL-2060s. The VAX 11/750 will be used primarily for geometry crunching, preparing MEBES files, and writing MEBES tapes.

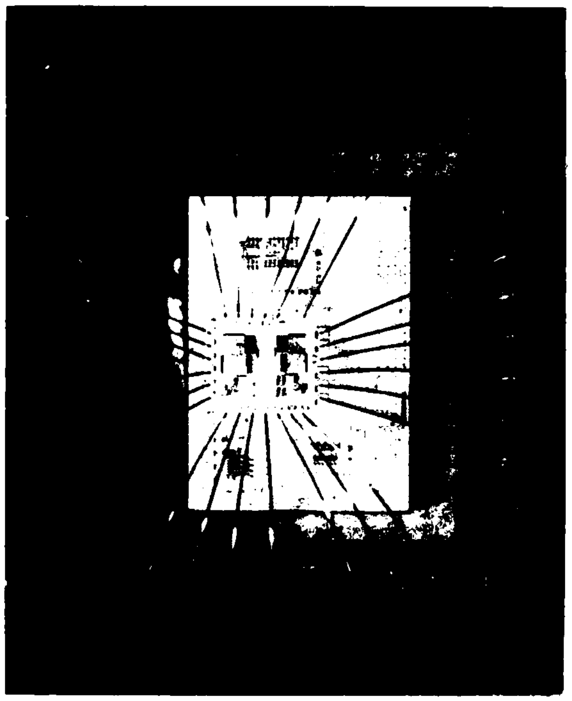
5.3.3 Fabrication Interface

The MOSIS vendor base has expanded substantially during this reporting period. Increased user feedback and more extensive test results have allowed the MOSIS project to determine and communicate fabrication requirements to new vendors. This has resulted in higher quality wafers and the development of consistently reliable vendor sources for mask making and nMOS fabrication.

MOSIS has instituted procedures to manage the vast amount of information inherent in dealing with a multi-vendor base. Many administrative tasks have been automated, including the maintenance of templates to determine fabrication requirements specific to vendor and technology (a single vendor often provides several fabrication technologies).

Three micron CMOS/Bulk presents many complications that do not exist in nMOS: fabrication requirements vary considerably from vendor to vendor. We have been largely successful in standardizing the design rules for the users by deriving nonstandard designs from user-supplied





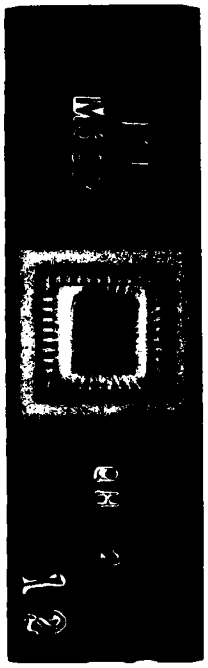


Figure 5-1: M3CR-Royal run: 3-inch nMOS wafer, 52 die types of 5 sizes containing 93 projects from 19 organizations

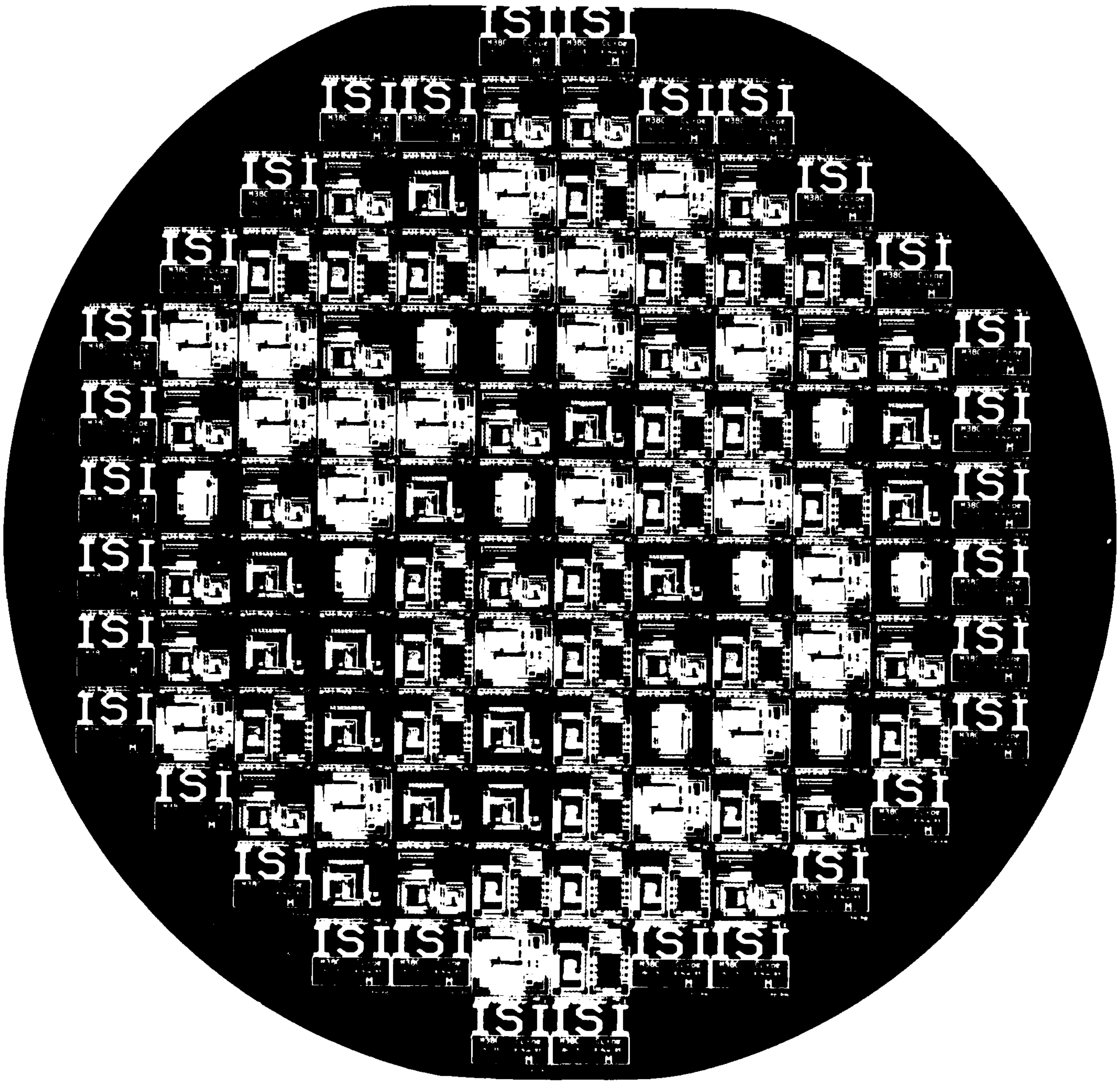


Figure 5-2: M38C-Clyde: 5-inch CMOS wafer 5 die types containing 8 projects from 3 organizations

designs, as in the case of extra layers. The availability of such fabrication options varies widely. For instance, a vendor may offer combinations of desirable options such as capacitors via an electrode layer and a second metal layer (capacitors, second metal, either but not both, or both). We anticipate that the standard 3 micron CMOS/Bulk technology will offer a choice of options, but not both. The availability of these options, coupled with increased volume and a diverse vendor base, necessitates continued automation of the fabrication interface.

The quality of parts fabricated in the CMOS/Bulk technology equals that of parts fabricated in nMOS. Further, CMOS/Bulk is a more desirable technology because of its capability for lower power consumption and higher circuit density. Therefore, MOSIS is preparing for the eventual shift away from nMOS by improving turnaround and vendor reliability for CMOS/Bulk fabrication.

5.3.4 Quality Assurance/Design Interface

Most MOSIS devices are prototypes without established functional testing procedures. Generally, the designers who receive these devices are still debugging the designs, rather than checking for fabrication defects introduced by less-than-perfect yield.

MOSIS's extensive quality assurance program is aimed primarily at the parametric level. This guarantees that the electrical properties of the wafers are within specifications established by the best a priori simulations used in the design process. Work has continued to increase the accuracy of the SPICE parameters which are made available to MOSIS users. SPICE provides simulated mathematical modes for behavior of transistors, allowing designers to assess a small digital circuit idea, to avoid faulty design, and to improve their chances of success in fabrication. The electrical criteria are a superset of the SPICE parameters at level II. They include a ring oscillator, which gives a rough idea of the speed of the resulting circuitry. The electrical properties of the wafers are extracted first by the fabricator, who uses either his own process control monitoring devices or the MOSIS test structures. Only wafers passing these tests are delivered to MOSIS.

It is a common practice in the IC industry to save functional probing time by probing wafers in only a very few sites. This practice makes sense, because all parts are subject to functional testing and because this parametric probing serves only to eliminate disastrously bad wafers.

Since designers hand-test most MOSIS devices, MOSIS requirements for parametric testing are higher than industry standards. MOSIS inserts its own test strip on every device, if space permits. When the test strips cannot be inserted, MOSIS probes a large number of test sites. This probing provides important statistics on the electrical properties and their distribution across the wafer. Most wafers have uniform distribution; some, however, have other statistical patterns, such as significant gradients and bimodal distributions.

These in-depth statistics are available only to the fabricators. Designers receive the general statistics (mean and variance) for each run. Interested users can request the specific values of the parameters extracted near any of their chips.

Users comparing the performance of actual chips to their simulations find it useful to rerun the simulation with the actual a posteriori parameter values extracted near that chip. The marking on each chip (which can be checked by opening the package lid and using a low-power microscope) identifies the run in which the chip was produced and its position on the wafer. Along with the wafer ID marked on the package, these markings provide the identification needed to determine the most relevant probing site.

A perfect set of electrical parameters does not guarantee perfect yield. so there is always a need for functional testing. MOSIS does not have the facilities for high-speed functional testing. but can perform partial functional testing. This screening typically catches the majority of fabrication defects, such as shorts. The screening is performed by applying and reading user-provided vectors to each device before the wafer is cut: those failing the test will not be packaged. By screening the larger chips—which typically have lower yield and higher packaging cost. and are required in larger quantities—MOSIS significantly reduces the packaging cost.

5.3.5 Standard Pad Frame/Packaging

MOSIS's current packaging strategy is to package enough parts to ensure a 90 percent probability of delivering a defectless part to the designer. This strategy was acceptable when most of MOSIS's community was designing small circuits and the fraction of packaged defective parts was small. However. a significant portion of the community has successfully completed the development of large designs and now wants from 300 to 3,000 working parts to begin developing prototype systems based on parts obtained through MOSIS. The yield for these large designs is expected to be 25 percent at best. If MOSIS were to follow its current strategy of packaging parts without any testing to indicate functionality, it would be packaging four times the required number of parts to achieve a requested quantity.

To avoid such waste, MOSIS has worked with Stanford University to define a functional test language (SIEVE) and has developed hardware to effect the testing specified by that language. Users now have the option of submitting text describing limited test procedures to be used at wafer probe to screen out bad parts. The purpose of this screening is to detect the types of "trivial" defects that cause the majority of bad parts and, therefore, to reduce packaging costs. Full functional testing is expected to be done by the user.

For designs with custom pad layouts, it is the responsibility of the designer to provide MOSIS with the custom probe card to probe his circuits. To eliminate the inconveniences associated with generating custom probe cards for every design, MOSIS has developed a set of standard pad frames, each specifying exactly where the pads are positioned. MOSIS stocks probe cards for each of the frames.

These standard frames are also expected to facilitate packaging. Bonding diagrams for projects currently submitted are generated manually. because several attempts to automate this process have met will only limited success. Bonding diagrams instruct the packager to connect a specific pad on the chip to a specific pin on the package. Standard pad frames have standard bonding diagrams. eliminating the the need to generate a new diagram for each project. The increased use of the standard pad frames has reduced the considerable amount of time needed to manually generate bonding diagrams.

Standard frames also allow the bonding process itself to be automated. Automated, programmable bonding machines are currently available. Standard pad frames make possible a scenario in which an operator would identify a first pad and package pin; programmed information would then control the bonding on a chip.

Automating the packaging phase could significantly improve MOSIS's turnaround time. The best times experienced by MOSIS for various parts of a run in nMOS are (1) four days to convert submitted geometry into a format acceptable by a mask maker and to generate the first masks required by a fabricator to start a run; (2) five days for wafer fabrication; (3) one day for wafer acceptance; (4) seven days for packaging; and (5) one day for overnight delivery of packages. Standard pad frames with automated bonding could reduce packaging time to two or three days and reduce the time from submission of designs to mailing of packaged parts (in very conventional technologies) to approximately two weeks.

5.4 IMPACT

MOSIS's main function is to act as a single interface ("silicon broker") between a geographically distributed design community and a diverse semiconductor industry. As such an interface, MOSIS has significantly reduced the cost and time associated with prototyping custom chips.

The greatest impact of MOSIS, however, is in the community it has created. The MOSIS user community shares not only the fabrication services, but also experience, cells, tools, and software. The rapid growth of the community proves that the services provided by MOSIS are useful and important to both the academic and the R&D communities.

5.5 FUTURE WORK

5.5.1 CMOS Cell Libraries

A large part of the MOSIS community is now designing in CMOS. In order to allow the community to build on top of the work it has already done, MOSIS will develop a library of community designs and make it available to new designers.

6. ADVANCED VLSI

***Research Staff*:**
George Lewicki
Ron Ayres
David Booth
Danny Cohen
Joel Goldberg
Lee Richardson
Barden Smith
Victoria Svoboda
Vance Tyree

***Research Assistants*:**
Yehuda Afek
Joseph Green
David Hollenberg

***Support Staff*:**
Kathie Fry
Terri Lewis
Jasmin Witthoft

6.1 PROBLEMS BEING SOLVED

Military systems are currently limited, not by technology, but rather by the difficulty systems designers have in accessing that technology. In the past few years, the MOSIS project has successfully interfaced a large VLSI design research community to VLSI technology available within the American semiconductor industry. The result has been some remarkable innovations in computer architectures. The tools used were large computer resources and access to the community being served by a computer communications network.

Considerable innovation in systems designed for the Department of Defense is expected to occur if the approach used in MOSIS (interfacing a DoD VLSI design community to fabrication technology) is used to interface a DoD systems design community to the technologies on which it depends. The result will be a drastic shortening of the turnaround time required to produce prototype systems.

The main technology on which systems are dependent is VLSI. The natural first steps for this project are the provision of interfaces to 1.2 micron CMOS bulk technology necessary for systems based on 100,000-transistor chips; access to fabrication of printed circuit boards (PCBs) containing both custom-designed and standard VLSI parts: and access to centralized functional testing of custom-designed parts prior to their assembly into packages and boards. It is these services which must be provided for system designers if they are to easily assemble prototype systems.

There is normally a considerable time lag between the availability of a technology and a large systems design community being able to design systems which best take into account the characteristics of that technology. The availability of a technology simply means that processes have been developed which can reliably generate large numbers of devices on silicon. The time lag has to do with circuit designers being able to develop circuit design techniques and system designers being able to develop architectures that best take advantage of the characteristics of the technology.

CMOS bulk technology with a feature size of 1.2 microns is currently being developed in the research departments of a number of commercial organizations. It is not expected to be generally available until late 1985. In the normal course of events, one would not expect system designers to be able to take full advantage of the technology until about 1988. The work performed by the Advanced VLSI project is expected to shorten this normal course of events by a full three years. It allows the technology to develop in parallel with design rules and circuit and systems design techniques.

A VLSI system is a collection of integrated circuits electrically connected in a printed circuit board. Development of prototype VLSI systems requires access to a service generating PCBs from parts and a specification. Before PCBs can be fitted with parts, there must be some degree of assurance that the parts are functional. Economy dictates that functional testing begin at wafer probe, prior to the packaging of parts. The Advanced VLSI project provides DARPA systems designers with PCB sources and functional testing.

6.2 GOALS AND APPROACH

The primary goals of the Advanced VLSI project are to provide the DARPA VLSI systems design community access to 1.2 micron CMOS bulk technology and to PCB fabrication: and to provide centralized functional testing of custom-designed parts prior to their assembly into packages and boards.

6.2.1 CMOS Bulk, 1.2 Micron

CMOS Bulk fabrication will be conducted in three phases. The first phase of this project involves a limited set of the DARPA research community submitting designs aimed at determining design rules applicable to a wide range of 1.2 micron bulk processes under development by U.S. industry. ISI will carry out all testing of parts in-house, with results rather than parts supplied to the experimenters. The reason for this is security: the commercial organizations currently developing 1.2 micron CMOS bulk processes hold these processes as very proprietary. and they will give the DARPA community early access to these processes only if security precautions are taken to prevent casual viewing of wafers. The only way in which such security can be guaranteed is to limit movement of wafers to the ISI environment. ISI has developed a test specification language for experimenters to use to specify tests to be carried out on their designs. ISI will use these test specifications to carry out tests on devices generated within the first phase of this program.

During the second phase, MOSIS will arrange fabrication runs for a selected segment of the DARPA research community. The primary purpose of these runs will be support of circuit design experiments for the development of circuit building blocks. Again, all testing will be performed at ISI and results will be distributed to the investigators. ISI will develop the necessary test languages and acquire the necessary test equipment to be able to carry out these measurements.

During the third stage of the project, when the technology has been fully developed, MOSIS will offer fabrication in this technology at the rate of one run per month, to the whole of its community. During this phase, public viewing of the actual circuits will be considered acceptable. and parts will be assembled and distributed to the designers. At that time. measurements for parametric and yield test structures developed within the first phase of the project will be used for the purpose of wafer acceptance.

6.2.2 PCB Services

The purpose of the PCB service is to support the packaging requirements of the research community for assembling integrated circuits into systems. MOSIS supports the production of printed circuit boards in a manner similar to the ongoing VLSI activities, by interacting with designers (using the MOSIS net-based user interface), accepting designs (written in the language CIF). and converting them into working parts.

MOSIS maintains design rules and recommends procedures for PCBs. and provides instructions to users.

As many tooling formats are used throughout the industry. several software systems have to be developed to allow for the use of a wide vendor base. Further, in anticipation of problems arising in the move to finer geometries (such as 0.005 inches and 0.003 inches on a 24-inch board). MOSIS is developing software for driving laser plotters which play a role similar to E-beam mask making devices for VLSI.

6.3 PROGRESS

6.3.1 CMOS Bulk, 1.2 Micron

As part of the first phase of this program. ISI obtained a set of design rules relevant to 1.2 micron CMOS Bulk technology from the National Security Agency and distributed them to the limited systems design community. Using this preliminary design set. the design community is currently engaged in developing a more workable set of design rules. They have also developed a geometry for test structures. to allow them to verify the design rules they are developing. In addition. ISI developed interfaces with four vendors (General Electric. Honeywell. Hughes Research Malibu. and National Semiconductor) capable of providing 1.2 micron CMOS Bulk fabrication.

The first round of designs has been submitted to ISI by the design community, and wafers have been returned from General Electric (two runs) and Hughes (one run). Fabrication is also in process at National and Honeywell: wafers will be returned to ISI in October 1984.

Automatic data-taking has been performed on the three completed runs. and we will return the results of the functional tests to the relevant designers and fabrication vendors. We are also planning to publish a book containing a synopsis of the data gathered from each of the test runs, organized by vendor.

The first phase of the program is proceeding on schedule. The fabrication and testing of the initial round of test structures will determine the reliability of the process and establish a working set of design rules. When this initial round is complete, the project will initiate the second phase of the program by allowing a somewhat larger set of the systems design community to submit actual circuit designs for fabrication. At the next DARPA meeting. in October 1984, ISI will distribute documentation detailing how these designers can submit their designs. As in the first phase, only test results will be returned to the designers; all testing will be done at ISI.

6.3.2 Scalable Design Rules

Designing integrated circuits at the 1.2 micron level will probably always be very expensive. because of the limitations of the technology. With larger feature sizes. several die types can be combined on the same wafer. allowing many projects to share the cost of fabrication. However. 1.2 micron circuits require the use of reticles. which are stepped across the wafers to define the geometry on the wafers. The necessity for reticles limits each wafer to one die type. thus limiting the number of projects that can be included on each run.

To avoid some of the expense involved in designing 1.2 micron integrated circuits, some members of the DARPA design community have suggested that circuits should be designed and perfected at a larger feature size and then "shrunk" to 1.2 microns for production. This process necessitates the development of a set of *scalable* design rules.

Don Trotter of the University of Mississippi has developed design rules that are scalable from 3 microns to 1.2 microns. The Advanced VLSI project is currently verifying that this set of design rules is acceptable to a major portion of the vendors who supply 3 micron CMOS Bulk fabrication. Once these rules have been accepted by the semiconductor industry, the Advanced VLSI project will suggest to the DARPA design community that they use this set of rules to design integrated circuits at the 3 micron size. When the 1.2 micron technology becomes available, these designers will be able to scale their designs down for fabrication at that feature size.

6.3.3 PCB Services

During this reporting period, MOSIS began to offer the fabrication of Printed Circuit Boards (PCBs) to its users. MOSIS treats PCBs as just another technology, surprisingly similar to nMOS and the various dialects of CMOS. Although PCBs and ICs are made of different materials, they share a common method of specification of the images required on each of their layers. Both are fabricated by a "photographic" process which transfers images to the media surface from a master tooling.

This approach of using common tooling preparation methodology for both ICs and PCBs has many advantages: it allows designers to use common tools for the design process (with details tailored specifically to each technology) and allows MOSIS to apply the same management procedures and the same geometrical processing and tooling preparation methods to PCB technology.

MOSIS has provided a total of 81 PCBs to nine designers since offering this *service; see Figure* 6-1 for an example.

6.4 IMPACT

The Advanced VLSI project was initiated in order to interface the DoD systems design community to the technologies on which it depends. By providing access to fabrication of custom printed circuit boards, performing functional testing of parts prior to packaging, and working to make new VLSI technologies available to this design community, the Advanced VLSI project is significantly reducing the costs and delays involved in the design of prototype VLSI systems.

6.5 FUTURE WORK

6.5.1 CMOS Bulk, 1.2 Micron

The Advanced VLSI project will soon initiate the second phase of the 1.2 micron CMOS Bulk program. During this phase, MOSIS will arrange fabrication runs to support the development of circuit design techniques and building blocks. All testing will be performed at ISI and results will be distributed to the investigators and the appropriate vendors. In addition, we will be distributing the scalable set of design rules to the entire design community so that they may begin designing circuits that can be scaled down to 1.2 microns at a later time.

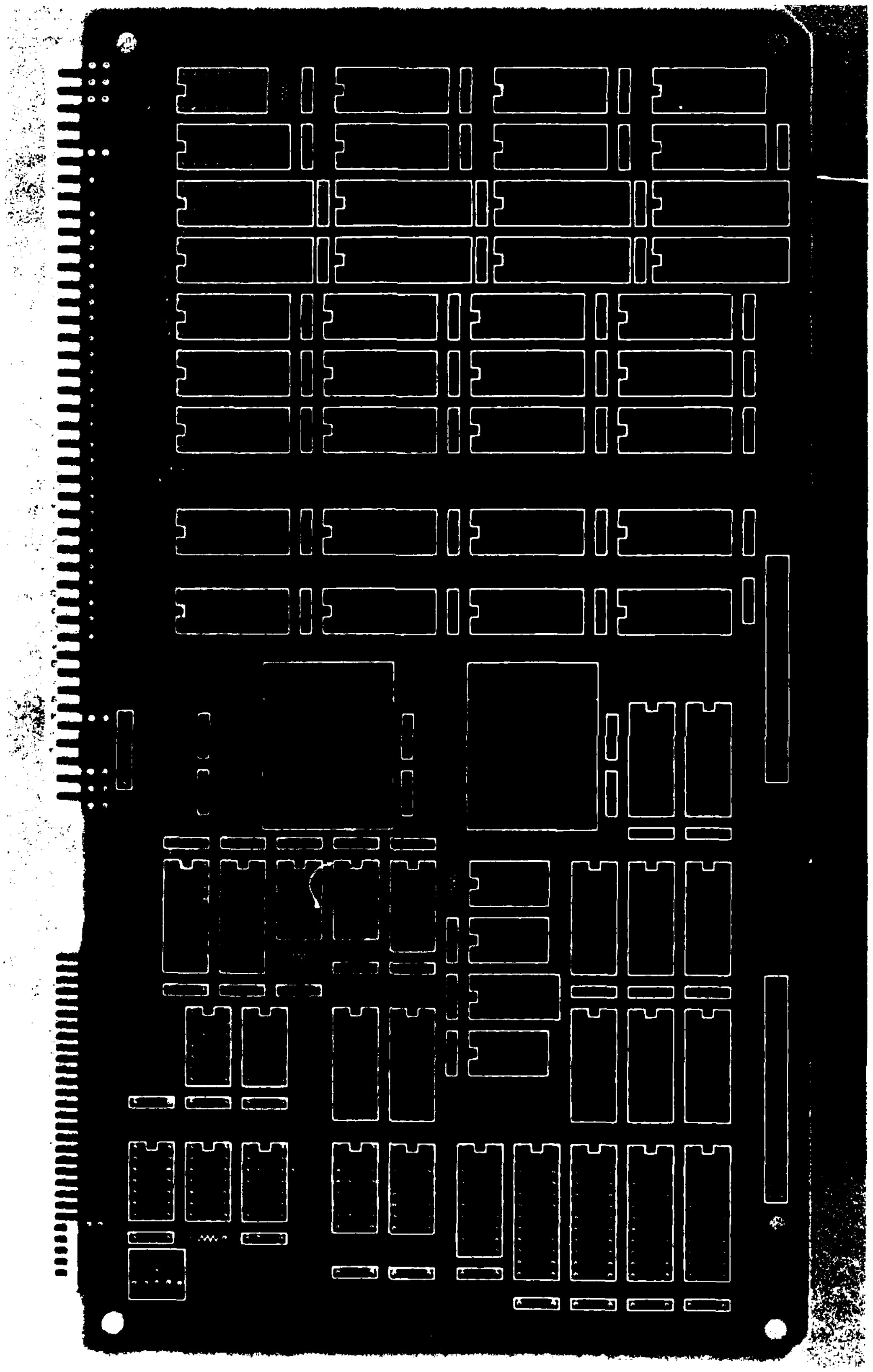


Figure 6-1: P45L: This PCB runs six times faster than the VAX-11/780 with floating point accelerator.

6.5.2 Packaging

We will investigate more advanced packaging techniques. such as various ceramic carriers (direct inking. etc.) and plastic tape carriers. ISI is developing the necessary interface (software, frames, instructions for users. etc.) to support the use of these advanced packaging technologies by the community.

6.5.3 Other Services

As the needs of the systems design community become apparent, the Advanced VLSI project will move to meet them. Specific areas of interest in the coming year are the construction and maintenance of cell libraries and pad libraries. and a pad router that will support 1.2 micron CMOS Bulk technology.

7. COMMAND GRAPHICS

Research Staff:
Richard Bisbey II
Eric Anderson
Benjamin Britt
Danny Cohen
Dennis Hollingworth

Research Assistant:
Elizabeth Lukasik

Support Staff:
Lisa Trentham

7.1 PROBLEM BEING SOLVED

The military, like its private sector counterpart, currently finds itself in the midst of an information explosion. More computers and computer-controlled systems are being acquired. generating information in ever-increasing quantity and detail. For this information to be useful in decision making, it is necessary that computers take more active roles in storing, retrieving, analyzing, integrating. and presenting data. For example. online computer file/information retrieval systems can supplant voluminous binders of paper. supplying more timely and accurate information.

When computers are involved in aiding the decision maker, the man-machine interface is a critical link. Information must be presented to the decision maker in ways that enhance and facilitate the decision process. Here. two-dimensional graphics can play an important role. Where spatial relationships exist, plotting the information on a graph. bar. or pie chart or positioning the information on a map can aid in the rapid assimilation of the information by the decision maker. Such two-dimensional displays can even disclose perspectives (e.g., a trend on a graph or a clustering of forces on a map) that would not be readily apparent from a table or list of numbers. Finally. two-dimensional displays provide a natural medium for integrating and fusing information. For example. a situation display application could graphically integrate surveillance. force, and meteorological data reported by dispersed forces to produce a coordinated picture of a particular situation. The resulting graphics display could then be used by the dispersed forces to coordinate their activities.

Three basic components are needed if graphics is to play a role in military decision making. First, there must be a graphics system, a program that converts basic graphics primitives (e.g.. lines, text. filled solids) into the order codes for a particular display device. This system must meet military requirements for mobility and survivability, particularly in crisis mode. The system must be adaptable to the changing communications. processing. and display resources available during and between crises. and must evolve to meet future command and control processing and display requirements.

Second. high-level tools are needed with which very complex two-dimensional graphics displays can easily be created. and which remove the burden of graphics expertise from the user. The tools would be in the form of intelligent. domain-independent display agents that. given a collection of data produce a graphics representation of that data. For example. an agent might produce a graph from a list of numbers or draw and annotate a map. The agents would use the graphics system for displaying the output.

Third. high-level applications and decision aids are needed. These applications and decision aids would interact with the user and the user's databases to help reduce the data to an understandable form and formulate alternative courses of action. They would perform the semantic binding between the user's data and the high-level graphics representations provided by an intelligent display agent. The decision aid, for example, would determine which high-level graphics representations would be used to denote a returning aircraft or a recovery site. The decision aid would also provide the geographic coordinates of the objects for placement on a map or chart. The application or decision aid would not be concerned with the particular techniques employed to support the various representations, nor with the production of those representations.

7.2 GOALS AND APPROACH

The Command Graphics project builds upon previous work performed by ISI for DARPA to make graphics more readily available and to enhance decision making capability in the military command and control environment. In a previous contract, ISI developed a graphics system for use in command and control applications. The system is both display–device independent and network distributable. The system's display–device independence permits graphic application programs to be written without regard to the particular type or location of the display device upon which the graphic output will ultimately be displayed. The system provides the application program with a set of generic, device–independent, two–dimensional graphics primitives by which pictures can be described and interacted with at the application program level. The system maps these generic primitives into the capabilities of the particular display device being used. A variety of application programming languages and output device types are supported.

The system was designed to be network distributable across multiple host computers. Distributability permits the graphics display device and operator to be located away from the host computer on which the application program is run, possibly at a small remote–site computer. Distributability allows the system to be flexibly configured to utilize available computation and communications resources, particularly important in crisis situations where resource requirements cannot be predicted. As originally implemented, the system could be distributed between PDP-10s, DEC System 20s, and PDP-11s connected by the ARPANET, Packet Radio, or Packet Satellite. This system has been used in the past as the backbone of Situation Display, an application-specific, natural-language–based, naval database query and display system.

During previous work, as part of the Situation Display and Assessment application prototyping effort, ISI also began to explore the features required for high-level tools to aid in the creation of geographically oriented displays. The Situation Display was a cooperative effort between ISI and several other DARPA contractors. ISI's contribution to this effort was the graphics output portion of the system. The Situation Display allowed an operator seated in front of one or more graphics display devices to pose natural language questions and commands to a distributed database and to receive natural language and graphics responses. Examples of natural language input included "Do any ships within 400 miles of Luanda have a doctor aboard?" and "Show me the destroyers whose radar is inoperative." Graphics responses could be in either of two forms, tabular or geographic. In tabular mode, basic ship information was displayed, e.g., identification code, name, type, class, speed, course, present track position, date of last reporting, and percentage of remaining fuel. In geographic mode, ships were positioned on a Mercator projection map. The operator could also display speed/course vectors, sensor envelopes, satellite footprints, sea lanes, and projected routes.

During the development of the Situation Display it was found that three specific display qualities were particularly important in developing aesthetic geographic-based displays: placement of objects, display precedence relationships between objects, and color selection.

The placement of information on the display proves to be critically important if the display is to be readable In creating maps and charts, cartographers go to great lengths to improve readability, even to the extent of displacing objects from their actual geographic coordinates. For example, consider a map of the United States showing major transcontinental highways and railways. In many instances, the two run parallel to each other. If both the highway and the railway were represented at their exact geographic locations, the two would lie on top of each other, making it difficult to distinguish one from the other A cartographer would displace one of the two some distance from the other so that both would be visible. In the Situation Display, symbol "collision avoidance" algorithms were included to perform this same function.

Another area addressed in the Situation Display was precedence of overlays. It is often necessary to call attention to large areas. These areas might represent some meteorological phenomenon, such as cloud or fog cover: coverage by sensing equipment, such as a satellite footprint or a radar sensor envelope: or a path, such as an airroute, an airway, or a sealane. While it is important to be able to visually distinguish the area in question, it is also important that the area be displayed in such a fashion as to avoid obliterating or obscuring other display information in that same area. Overlay precedence allows one to denote these areas by making small changes to either the intensity or chromaticity of other objects being displayed in that area, somewhat like placing a light-colored piece of cellophane over the area. The Situation Display used overlay precedence in displaying satellite footprints, sensor envelopes, and sealanes. For example, satellite footprints could overlay sensor envelopes, which could in turn overlay ships, sea, or land. All were discernible.

Color also had important use in Situation Display For ship information, color provided discrimination between nationalities. Highlighting was used to draw special attention to individual ships (e.g., for responding to questions such as, "Of the ships being displayed, which ships have an ASW capability?"). Colors and intensities were chosen so as not to unduly emphasize one type of object over another. Colors for background information were chosen to produce a visually apparent background. The same was true for precedence overlays.

Two important observations were made from the previous work. First, the production of legible geographic-based displays requires a considerable amount of graphics intelligence and sophistication. In fact, the amount of programming necessary to incorporate the "graphics intelligence" often exceeds that of the application program itself.

Second, the representation mechanisms used in Situation Display were essentially generic in nature For example, the same annotation mechanism used to represent the location of a ship or submarine in Situation Display could also represent the location of a tank or gun emplacement on an Army Tactical display or an aircraft being recovered in an Air Force Bomber Recovery display. While the graphics display portion of Situation Display was highly tailored to a particular application domain (pictures were described in terms of specific Navy objects, e.g. Vessel Control Numbers, Unit Identification Codes, Ship Classes), if the binding of semantic information to graphic representation was done in the application program the same graphics display package could be used by a variety of different applications. The major benefit of such a strategy would be to localize the "graphics display intelligence" in a single module that would be implemented only once for all applications rather than once per application.

Utilizing the existing graphics system as the underlying mechanism, an effort was initiated to design and develop a high-level "intelligent" agent that automated many of the functions a cartographer normally performs when creating geographically oriented displays The agent would assume total responsibility for the production of the display, including generation of background maps and generation, placement, and symbol collision resolution of user annotations. It would provide the application programmer with a high-level object-oriented interface for constructing geographic-based displays. Thus, the agent would reduce the graphics expertise required for developing aesthetic displays and would permit the programmer to focus on application issues for which he was most qualified. The agent's primitives would include maps, display and placement precedence control, and a general annotation mechanism, as well as traditional graphics primitives such as text, vectors, areas, lines, and swaths. Coordinates would be specified in latitude and longitude, with the agent assuming responsibility for the transformation to screen coordinates of the particular map projection being displayed.

7.3 SCIENTIFIC PROGRESS

During the reporting period, substantial effort was directed toward producing the intelligent agent described above, hereafter referred to as the Geographic Display Agent (GDA). The architecture of the overall system was defined, the programmer interface to the system was specified, a design effort to identify and relate specific system components was undertaken and completed, and a partial implementation of the GDA was completed. This version of the GDA allowed the application to specify a desired map area and place icons with associated textual annotations on that background map.

The GDA provides the C2 application programmer with an object-oriented interface that automates many of the functions a cartographer normally performs when creating geographically oriented displays. Its purpose is to allow the application programmer to concentrate on deciding which objects to display rather than the mechanics of actually generating an aesthetic display. The GDA handles issues such as insuring that alphanumeric information does not overwrite and thus obscure other alphanumeric information; managing the display precedence of objects to avoid obscuring important display information; managing object placement order, so that those objects considered most important are placed closest to their desired position; and transforming geographic coordinates into screen coordinates on a background map. The user can specify that a symbol with a specific label be positioned at a particular latitude and longitude, without having to perform the mathematics to calculate where it would be located on a Mercator Projection map. If there is already a symbol at that location, the GDA will relocate the one with the lowest placement precedence to an available screen location.

The user or his application program generates a sequential text stream that specifies the desired graphics display. This specification includes title, classification, and date; a background map area; and a list of the desired graphics with their intended map coordinate locations. The GDA processes the textual specification, placing and drawing symbology on the requested background map according to various positioning criteria, and creates an aesthetic, human-readable display representation tailored to the connected display device.

The GDA supports four basic kinds of graphic objects: markers, tags, linear objects, and areas. A particular object can have a variety of attributes that determine when, where, and how the object is to be displayed. These attributes include collision avoidance, placement precedence, display precedence, locations (in geodetic coordinates), and color. The GDA includes an automatic collision avoidance mechanism. This mechanism prevents certain object types from being written on top of and obscuring one another, thus reducing local clutter on the display and enhancing overall display intelligibility. The GDA also includes an integrated map system to provide background maps. The map system constructs maps of any part of the world upon demand from a specially constructed map database.

During the reporting period, the porting of the graphics system to the VAX/UNIX environment was completed. Conversion of the graphics system to the VAX/UNIX environment and reimplementation of the graphics system in the language C was established as a major step in the porting of GL to a microprocessor environment. The ultimate objective is to provide an inexpensive, self-contained, highly portable graphics capability for remote users such as those located in a HERT (Headquarters Emergency Relocation Team) van or an airborne command post. Upon delivery of a SUN/68000-based processor to ISI, the process will be completed with the porting of the VAX-based version of GL to the Motorola 68000-based processor running UNIX. In addition to the porting of the graphics system, various graphics applications and databases were rewritten in C and ported to the VAX. These included the Briefing System, the Map Drawing Package and its tiled, direct-access database, and the Interactive Graphics Language System.

7.4 IMPACT

The principal impact of this work will be felt in military environments where command mobility is paramount and a portable, network-based graphics capability is important to the command decision process. Completion of the conversion process makes the Graphics System available in the HERT van and other mobile environments. Development of the Geographic Display Agent facilitates development of decision aids and other application software having a strong, geographically oriented information presentation requirement. The work also serves as a model for future tool development by clearly demonstrating the utility of developing flexible and adaptable tools that are not tied to specific hardware and that incorporate intelligence about particular areas of functionality to be supported.

7.5 FUTURE WORK

Future work will address three areas: the Geographic Display Agent, Maps, and the graphics system.

The GDA work will focus on completing the initial implementation and then extending it. The most significant extension will be the inclusion of graphic input to allow the user to interact graphically with a C3 application by pointing to or touching relevant objects. Other extensions will include support for user-definable icons; inclusion of complex multi-element tags, such as labeled pie charts and gauges; and support for foreground transparencies/overlays.

The Map work will focus on alternative storage media for high-resolution maps. As part of this effort, we will incorporate a video disk map database into the GDA to produce a representative integrated multimedia map environment.

The graphics system work will focus on porting the graphics system to a a Motorola 68000-based terminal such as the SMI SUN terminal.

8. INTERNET CONCEPTS RESEARCH

***Research Staff*:**
Jon Postel
Danny Cohen
Paul Kirton[1]
Paul Mockapetris
Greg Finn
Ruth Brungardt
Annette DeSchon
Joyce Reynolds

***Research Assistants*:**
Alan Katz
Dave Smallberg
Marshall Rose

***Support Staff*:**
Ann Westine
Monica Boseman

8.1 PROBLEM BEING SOLVED

The goal of the Internet Concepts Research project is to extend and enhance computer communications. The project's work is based on the ARPA-Internet, a system of interconnected computer communication packet networks. The rules of communication (or protocols) are the key element in successful computer communication. The ARPA-Internet has working protocols for communication between heterogeneous computers via an interconnected collection of heterogeneous packet networks. The ARPA-Internet currently has 933 hosts, 158 networks, and 89 gateways.

This research covers work at several levels, involving applications protocols as well as host-to-host and gateway-to-gateway protocols. In the applications level protocols, the focus is on new uses of the Internet based on new procedures for abstract process communication, hierarchical naming, multimedia mail, high-speed network interfaces, and new capabilities in the gateway protocols. In the gateway and host level protocols, these extensions include mechanisms and procedures for monitoring and maintaining performance and survivability, for allowing continued growth of the Internet, and for dynamic control of the Internet. The basic protocols are largely complete, but a number of extensions are being explored to integrate and extend the packet network technology.

8.2 GOALS AND APPROACH

The long-term goals of the Internet Concepts Research project are to provide appropriate and effective designs for the primary user service applications in the internetwork communication environment. The designs are based on a set of host and gateway level protocols that provide the full range of service characteristics appropriate to a wide variety of applications.

Our approach has been to pursue in parallel the abstract design and experimental implementation of protocols. The interaction of these activities provides valuable insights into problems and potential solutions.

Further, we divide our effort into several task areas: Abstract Process Communication, Hierarchical Naming System, Multimedia Mail, High-speed Interfaces, and Studies, Surveys, and Specifications.

[1]Visiting Scholar supported by Telecom Australia

8.2.1 Abstract Process Communication

Computer communications systems to date have provided effective yet primitive means for process-to-process communication. Significant improvements in multi-machine applications will require more abstract and specialized process-to-process communication services.

Our goal is to develop a high-level abstract, internet-based process-to-process communication facility. Our approach is to study existing interprocess communication mechanisms, especially those used in distributed systems, and to incorporate the best features of the existing mechanisms into an internet process-to-process communication protocol.

8.2.2 Hierarchical Naming System

In a large internet system, the management of names is a complex task. The Domain Naming System is being developed to simplify this task; it provides structured names and allows subdivision of name management duties.

Our goal is to provide a design for a distributed system of Domain servers that manage the access to a distributed database of names and associated data. Our approach is to create the design for the system and to review it with interested parties in the DARPA research community. As a result of feedback received from these reviews, the design will be modified. Also, we will implement a prototype Domain server. The experience from this implementation will be used to modify the design and produce the final specification.

8.2.3 *Multimedia Mail System*

We have chosen computer mail as the application to use as a focus for demonstrating internetwork service. Within this application area, we are developing the Multimedia Mail system.

The Multimedia Mail system is an experiment to explore the future direction of computer mail systems. The primary goal is to provide specifications for computer-oriented data structures to communicate various types of data in messages, including text, graphics, and voice. Our primary interest in this mail system is in the communication mechanism, rather than the user interface.

8.2.4 High-speed Interfaces

The speed of local computer networks has increased more quickly than the speed of computers themselves. Our goal is to find ways of speeding up the interface between computers and local networks to the point that computers can intercommunicate at network speed. Our approach is to redesign the network interface to perform simple and frequent operations in hardware and more complex operations with software support, independent of protocol level.

8.2.5 Studies, Surveys, and Specifications

While the Internet system is now operational, there are many potential extensions which have been discussed and desired but which have not been designed, implemented, or tested. Our goal is to study some of these issues and, when possible, provide designs for their eventual development.

The areas of addressing, routing, and multiplexing are particularly subtle and require careful attention. We have concentrated on these areas and have explored many options in technical discussions and memos. Our approach is to develop an understanding of these technical issues and to advocate the inclusion of general-purpose support mechanisms in the protocol specification.

In the operation and the evolution of an internet system, it is sometimes useful to survey or measure the implementation status and performance of particular protocols. As the Internet system evolves, and as flaws and ambiguities are discovered, specification documents must be upgraded.

8.3 SCIENTIFIC PROGRESS

The major effort this year has been on Multimedia Mail and the Domain Naming System. The Abstract Process Communication and High-speed Interface tasks were study areas during this report period, and significant development in these areas will begin next year. Our continuing effort in Studies, Surveys, and Specifications produced several useful tools, reports, and specifications.

8.3.1 Abstract Process Communication

The goal of this effort is to develop protocols which will reduce the cost of constructing multicomputer applications. Developing these higher level protocols and associated abstractions is important if we are to create applications that span new personal workstations, specialized servers, and the existing general-purpose timesharing systems that currently make up the bulk of the Internet system.

The components of the solution include facilities for initiating processes on different machines on the Internet, facilities for interprocess communication, and facilities for controlling and monitoring the progress of a multicomputer application.

Present high-level communications services of the Internet (e.g., TELNET, FTP, MAIL) illustrate the problems faced by a system builder when a multicomputer application is built using today's technology. Although these services are multicomputer applications that use forms of interprocess communication (IPC), internally they neither share abstractions about IPC nor provide interfaces that calling programs can conveniently use to control them. Thus there are few instances of services built upon the existing high-level services, and essentially no instances of user-level multicomputer applications.

In the Internet, we expect to see both direct ownership of a process (i.e., a master-slave relationship) and cooperation between mutually suspicious peers (e.g., a compiler process accessing a file server process or databases exchanging updates). The master-slave type of interaction is supported through protocols that allow complete control of a subservient process, while the peer relationship is supported with protocols that processes can use to negotiate mutually acceptable parameters.

This year we began this task as a study area. We are reviewing the literature on interprocess communication.

8.3.2 Hierarchical Naming System

Last year we began the design of the Domain Naming System, which provides a hierarchical subdivision of the name space to allow an administrative subdivision of the task of maintaining the large and rapidly changing database.

The Domain System uses structured names to reference things (such as hosts and mailboxes) in the Internet. The resolution of the names to addresses is accomplished by accessing a Domain server.

The user program issues a query via a call to a local program known as a resolver. This query is expressed according to local conventions, and the resolver is usually part of the host operating system. The resolver answers the query using information it acquires from one or more name servers. The transaction between the resolver and the name server is accomplished using the Domain protocol.

Domain servers are respositories for sections of the Domain database. A given name server will typically have information for only a small part of the abstract database. Domain servers internally break the abstract database into sections called zones. While Domain servers can be configured to treat each Domain name as a separate zone, system administrators will usually configure Domain servers to group organizationally related Domain names in a single zone. A particular zone may be replicated at several Domain servers to provide higher availability (see Figure 8-1).

This concept was reported to the DARPA research community [3]. During this reporting period, we continued definition of the Domain System and issued preliminary documentation on Domain Names (including plans and schedules, concepts and facilities, and implementation and specification) [14,15,16]. In addition, we worked on the implementation of the Domain database, the implementation of Domain server and resolver; revised certain Domain specifications; and published a proposed Domain implementation schedule [20,22].

8.3.3 Multimedia Mail System

The system model asserts that a message service can be divided into two activities: message reading and composition, and message delivery. Message reading and composition is an interactive activity in conjunction with a User Interface Process (UIP). The message delivery activity may be carried out by background processes called Message Processing Modules (MPMs). Our work concentrates on the message delivery aspects and leaves the development of sophisticated user interfaces to other projects.

The Multimedia mail system is concerned with the delivery of messages between MPMs throughout an interconnected system of networks. It is assumed that many types of UIPs will be used to compose and display messages delivered by MPM processes. The MPMs exchange messages by establishing a two-way communication path and sending the messages in a tightly specified format. The MPMs may also communicate control information by means of commands

A message is formed by a user interacting with a UIP. The user may create various fields of the message and may invoke an editor program to correct or format some or all of the message. Once the user is satisfied with the message, he "sends" it by placing it in a data structure shared with the MPM. The MPM takes the data, adds control information to it, and transmits it. The destination may be a mailbox on the same host, a mailbox on another host in the same network, or a mailbox in another network.

The MPM calls on a reliable communication procedure to communicate with other MPMs. In most cases, this is a transport-level protocol such as the TCP. The interface to such a procedure typically provides calls to open and close connections, and to send and receive data on a connection. The MPM receives input and produces output through data structures that are produced and consumed by UIPs or other programs.

The MPM transmits the message, including the control information, in a highly structured format using typed data elements in a machine-oriented yet machine-independent data language [1,2].

Last year, the second version of the MPM for TOPS-20 was completed and tested on systems at ISI, Bolt Beranek and Newman, Inc., SRI International, and the Massachusetts Institute of Technology. This year, the MPM has been constantly available on USC-ISIF and USC-ISIB, with careful and continuous monitoring. This monitoring is intended to point out deficiencies in the MPM when run as a server with WHEEL capability. The MPM has been providing reliable regular service for the users of the experimental system (see Figure 8-2).

This year, a FORWARD/FORWARDING message pair was designed and implemented for the MPM. The FORWARD message is sent from an end-point MPM to a relay MPM which is known to hold messages for the end-point. Messages are held when the end-point is inaccessible to the Internet. The relay responds with a FORWARDING message to the end-point MPM telling it how many (if any) messages it has held for the end-point. The relay soon sends those held messages to the end-point MPM.

The Multimedia Mail program is used on a regular basis between three Perqs via two TOPS-20 systems within ISI (see Figure 8-3). Tests and demonstrations of the Multimedia Mail capability were also conducted for interested visitors. Multimedia Mail is also exchanged on a regular basis between ISI and Bolt Beranek and Newman, implementing text, bitmaps, and voice within the messages. Documentation was written on the Multimedia Mail User Interface program on the Perq [5], and a conference paper was presented on the Multimedia Mail system [27].

Programs are available to manipulate bitmap images on the Perq and on TOPS-20. A facsimile format conversion program, which converts between RAPICOM format fax files, CCITT format fax files, and bitmap format files were completed and implemented. It enables images to be entered into the system via the RAPICOM-450 facsimile machine or to be printed on that machine from files stored on the system [6,7].

Work has begun on a new implementation of the UIP on the Xerox 8010 using Interlisp-D. Evaluations are being carried out on the differing approaches for their "appropriateness" to the Interlisp environment. Experimental multimedia messages from the Xerox 8010 via the LISP multimedia message program have been successfully sent over the Ethernet to the MPM for delivery to other sites.

A program was written for the Perq (using ST and IP/ST protocols) to send and receive voice data to and from the packet voice terminal (PVT) via the Internet, allowing the PVT to be a speech device for the Multimedia Mail systems on the Perq. Work continued on the software to make the Multimedia UIP more robust and to play voice in the Perqs on the PVT over the phone system. Speech can now be recorded, played back, and sent within the Multimedia Mail system. The PVT can also be used as a voice server from the Perqs (see Figure 8-4.)

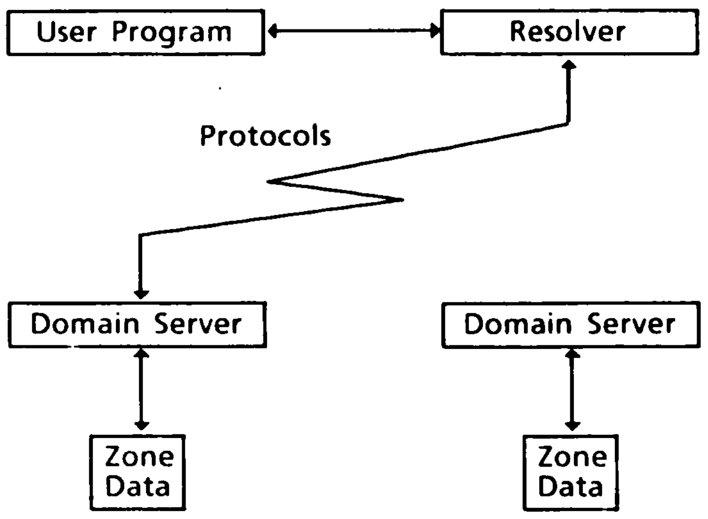


Figure 8-1: The Domain System

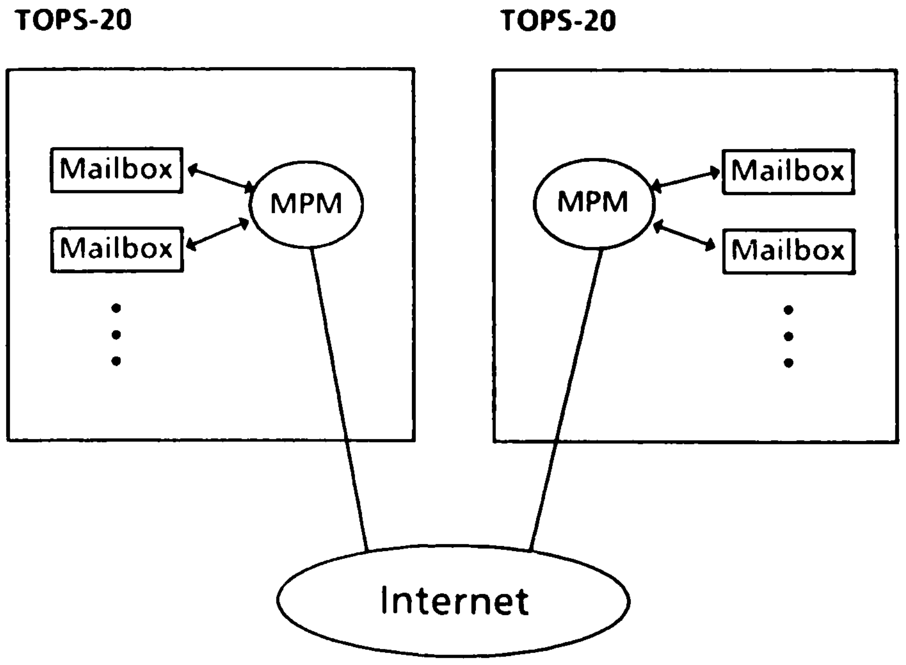


Figure 8-2: Mail is routed from source to destination mailboxes by the MPMs.

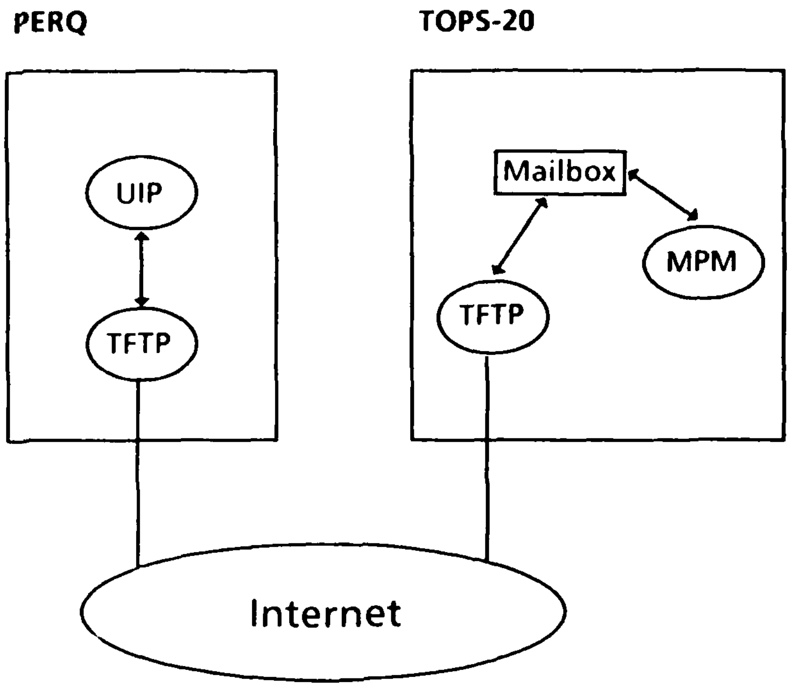


Figure 8-3: The UIP access the mailbox via TFTP.

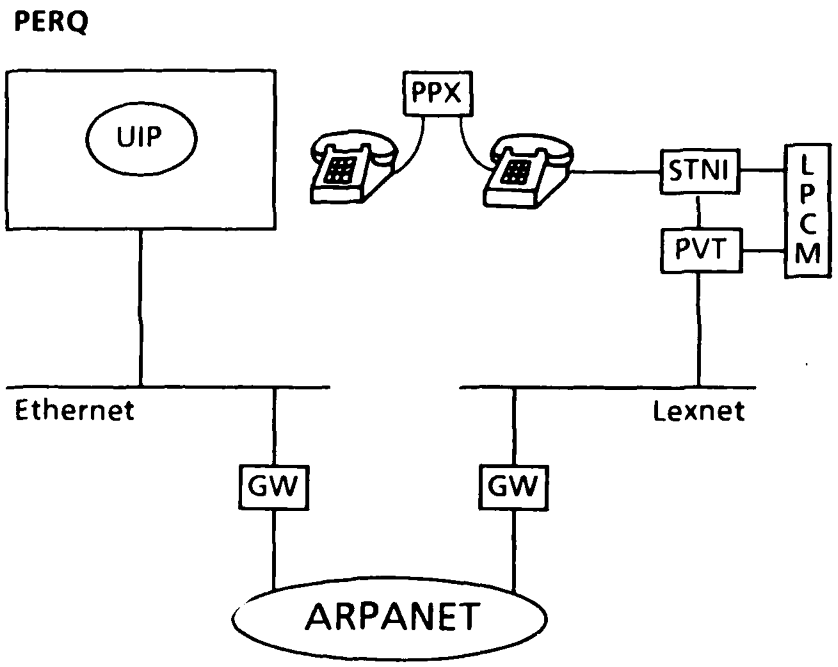


Figure 8-4: Voice I/O using an IP/ST/NVP server

8.3.4 High-speed Interfaces

This effort is concerned with developing specialized architectures for communications support. with the objectives of increasing the performance of communication and providing a natural environment for implementing protocol software. Our aim is to create implementations of standard protocol services. such as TCP and IP. that are as fast as the links in local networks. satellite channels. and other high speed communication links.

We continued to study local network communications from the host's point of view. We had previously determined that few protocol systems and interfaces allow the hosts to utilize the network at full speed. Typically. delays in processing prevent a host from sending or (especially) receiving back-to-back packets on the network.

We explored design concepts for a high-speed protocol interface. Our resulting design is called the "filter" interface [4]. We continued to explore and work on the design of this filter interface. including development of a TCP measurement routine.

8.3.5 Studies, Surveys, and Specifications

A Mail Relay program was developed on TOPS-20. to which access the MCI-Mail system via a dial-up modem. The program. MCIMailer. reads each message waiting in the MCI-Mail user's "in-box" into a TOPS-20 file. These files are reformatted to be ARPA Mail unsent mail files, and delivered to an ARPA-Internet user by one of the TOPS-20 mail programs. After MCIMailer has "read" the MCI messages. it picks up TOPS-20 files containing messages destined for MCI-Mail users. and uses the header information and the text from each of these files to "create" and "send" each message as MCI-Mail. Draft documention on this program has been completed for publication [32].

A Message Forwarding Program which runs on TOPS-20 and forwards short messages via a dial-out modem to the Meta System computer was implemented. These messages are then broadcast by the Meta System radio station. The messages are received by the pocket-sized beeper corresponding to the addressee in the message.

In support of research on Autonomous Systems. an experimental version of the Exterior Gateway Protocol (EGP) was implemented. The C-Gateway and EGP C Code from MIT were studied in preparation for adapting them to run on a VAX 11/750 at ISI. Autonomous System topology issues and backup gateways were studied. as well as the UNIX networking and routing code. in order to determine the best way to implement this version of EGP. The protocol functions and the routing part of the EGP user process were tested and adjusted to conform with the EGP specifications. The program is available, and is in use in 20 gateways throughout the United States.

We conducted a survey of the state of implementations of SMTP on Internet hosts [10]. This survey checked the connectivity from class A and class B hosts. the implementation of the VRFY command. the implementation of the mailbox "postmaster". and the treatment of an unknown mailbox. We completed the implementation of the TCP-based and UDP-based version of the Little Servers on a VAX-750 running Berkeley Unix 4.1.

An Internet "Spytool" was developed in the Mesa Development Environment on a Xerox 8010 workstation This tool may be used to capture and examine packets traversing an Ethernet. It has been used to diagnose protocol implementation problems.

We issued protocol specifications and memos on reference models and architectures, TCP maximum segment size. Telnet end of record. and transmission of IP datagrams over Ethernets [9,13,18.19,21].

We have continued to support the Internet Research Group through presentations. preparation of meeting notes, agendas. and other routine documents [11.12.23,25.29.30.31.32]. We also coordinate the monthly report for the DARPA Internet Program.

8.4 IMPACT

The impact of packet-switched computer communication has been enormous. The success of the Defense Data Network (DDN) is one example of the importance of this work. Further enhancements of such systems depend on continuing research in computer network technology. The work reported here extends the previous capabilities. The Multimedia Mail effort explores new user-to-user information exchange capabilities. The Domain Naming System extends the power of the Internet system to identify resources by name by increasing the flexibility and dynamics of name management. Abstract Process Communication creates a basis for much more powerful multi-machine applications. The provision for high-speed interfaces may permit the development of a new class of applications. The expanding use of the internet system requires a continuing effort to provide up-to-date knowledge of the system design, specifications. and performance.

8.4.1 Abstract Process Communication

The Internet system has been very successful in providing the current application services (remote terminal access, file transfer, and computer mail). In the future, as powerful workstations become more prevalent, new modes of communication and resource sharing will evolve. These will require much more complex multi-machine interactions. To implement such systems, a powerful set of basic interprocess communication facilities will be needed. These facilities must be at a substantially higher level of abstraction than is provided by the current TCP connections and UDP datagrams.

Plans are underway for several new applications that require substantial multi-machine intercommunication. Abstract Process Communication could provide a basis for such efforts.

8.4.2 Hierarchical Naming System

The implementation of the Domain Naming System will significantly ease the administrative burden on the current name management group. It will also place the control of changes and updates to the database closer to the organizations affected.

8.4.3 Multimedia Mail

The potential for multimedia communication in a computer-assisted environment is great. The ability to communicate diagrams or maps and to then talk about them will tremendously increase the effectiveness of remote communication. The combination of text. speech. graphics. and facsimile into a common framework and data structure may have substantial impact on other applications as well.

Computer mail is the most significant use of the new communication capability provided by packet-switching networks. Our continuing work to extend the range and capabilities of computer mail will have important consequences for DoD.

The power of a communication system is directly related to the number of potential communicants. For computer mail, this means that the power of a system is related to the number of people who have access to that system. Access to a computer mail system requires the use of compatible components: terminals, programs, and protocols. Our work on protocols and programs will increase the power of computer mail by enlarging the set of compatible components.

8.4.4 High-speed Interfaces

The demonstration of a high-speed interface could change the approach to implementation of layered protocols in computer systems.

8.4.5 Studies, Surveys, and Specifications

The selection of the IP and TCP protocols by DoD as the basis for a DoD internetwork protocol standard shows the impact of the work of the DARPA community on DoD communication systems. The development of the Defense Data Network (DDN) by the DCA is a major use of these protocols in an operational military system. This influence is demonstrated further by the mounting requests for information about IP/TCP from companies interested in producing commercial products or bidding on government contracts.

Through our participation in discussions at the Internet Working Group meetings and in technical meetings with other contractors, we have successfully influenced the development of many protocols and protocol features. Our publication in conferences, journals, and newsletters extends the impact of this work to others who design similar systems [8,9.17.22,24,26,28].

8.5 FUTURE WORK

We will continue to work in the five task areas: Abstract Process Communication, Hierarchical Naming System, Multimedia Mail, High-speed Interfaces, and Studies, Surveys, and Specifications.

8.5.1 Abstract Process Communication

- Study a representative sample of existing local interprocess communication mechanisms.
- Develop a draft Internet IPC Protocol, work with other DARPA contractors to refine this draft, and specify a final Internet IPC Protocol.
- Draft a preliminary standard for the remainder of the Abstract Process Communication system.
- Create a demonstration implementation of the IPC protocol.

8.5.2 Hierarchical Naming System

- Further develop and implement the Domain server program and supporting services and protocols.
- Create demonstration implementations on TOPS-20 and the Xerox 8010. The TOPS-20 implementation will be written in PASCAL and will include both the server and the

resolver functions. The Xerox 8010 implementation will be written in Mesa and will include only the resolver function. This resolver will communicate with the TOPS-20 server via the Internet.

8.5.3 Multimedia Mail

- Continue to operate the MPM program.
- Develop a simple user mail interface program on the XEROX 8010.
- Experiment with various styles of preparation of multimedia messages, including various editing techniques, the granularity of media elements, and voice implementation in the composite messages.

8.5.4 High-speed Interfaces

- Develop the design of the high-speed filter communication interface. Our eventual aim is a VLSI implementation of the filter. We will conduct software experiments to guide the design decisions.

8.5.5 Studies, Surveys, and Specifications

- Continue to conduct studies of Internet protocol implementation features and report the results to the ARPA-Internet community.
- Continue to conduct surveys and specifications of Internet protocol implementation features, conduct numerous performance measurements, and report the results to the ARPA-Internet community.
- Produce up-to-date documents of the protocols used in the ARPA-Internet community on an as-needed basis, and manage the assignment of protocol parameters to network experimenters as needed.

REFERENCES

1. Postel, J., "Internet Message Protocol," USC/Information Sciences Institute, RFC 759, August 1980.

2. Postel, J., "A Structured Format for Transmission of Multimedia Documents," USC/Information Sciences Institute, RFC 767, August 1980.

3. Su, Z., and J. Postel, "The Domain Naming Convention for Internet User Applications," Network Information Center, SRI International, RFC 819, August 1982.

4. Mockapetris, P., *Communication Environments for Local Networks*, USC/Information Sciences Institute, RR-82-103, December 1982.

5. Katz, A., "MMM - A Multimedia Mail System User Interface," USC/Information Sciences Institute, July 1983.

6. DeSchon, A., "Facsimile Support Programs User Guide," USC/Information Sciences Institute, September 1983.

7. Reynolds, J., "How to Create and Print a Bitmap Using a Hardcopy Picture," USC/Information Sciences Institute, August 1983.

8. Cohen, D., and J. Postel, "Gateways, Bridges, and Tunnels in Computer Mail," in *Local Net 83 – Strategy and Systems*. London, Online Conferences, March 1983. Also in *Local Net 83 – Distributed Office and Factory Systems*. New York, Online Conferences. September 1983. Also available as USC/Information Sciences Institute, RS–83–117. January 1984.

9. Cohen, D., and J. Postel, "The ISO Reference Model and Other Protocol Architectures," in *Proceedings of the Ninth World Congress of the International Federation for Information Processing*, Paris, September 1983. Also available as USC/Information Sciences Institute, RS–83–6, November 1983.

10. Smallberg, D., "Survey of SMTP Implementations," USC/Information Sciences Institute. RFC 876, September 1983.

11. Reynolds, J., and J. Postel, "Assigned Numbers," USC/Information Sciences Institute. RFC 870. October 1983.

12. Reynolds, J., and J. Postel, "Official Protocols," USC/Information Sciences Institute. RFC 880, October 1983.

13. Postel, J., "The TCP Maximum Segment Size and Related Topics," USC/Information Sciences Institute. RFC 879, November 1983.

14. Postel. J., "The Domain Names Plan and Schedule." USC/Information Sciences Institute, RFC 881. November 1983.

15. Mockapetris. P., "Domain Names – Concepts and Facilities," USC/Information Sciences Institute. RFC 882. November 1983.

16. Mockapetris, P., "Domain Names – Implementation and Specification," USC/Information Sciences Institute, RFC 883, November 1983.

17. Mockapetris, P., "Analysis of Reliable Multicast Algorithms for Local Networks," in *Proceedings of the Eighth Data Communications Symposium*, ACM/IEEE, Falmouth. Massachusetts, October 1983. Also available as USC/Information Sciences Institute. RS–83–10, November 1983.

18. Postel. J., "Telnet End of Record Option." USC/Information Sciences Institute. RFC 885. December 1983.

19. Postel. J., "Exterior Gateway Protocol Implementation Schedule." USC/Information Sciences Institute. RFC 890. February 1984.

20. Postel. J., "Domain Name System Implementation Schedule." USC/Information Sciences Institute. RFC 897. February 1984.

21. Postel, J., "A Standard for the Transmission of IP Datagrams Over Experimental Ethernet Networks." USC/Information Sciences Institute. RFC 895. April 1984

22. Reynolds. J., and J. Postel, "Domain Style Names Adopted by Internet." *CSNET News*. Published by CSNET CIC. Number 4. Spring 1984

23. Hinden, R., J. Postel, M. Muuss, and J. Reynolds. "Gateway Special Interest Group Meeting Notes." USC/Information Sciences Institute. RFC 898, April 1984.

24. Mockapetris, P.. J. Postel, and P. Kirton, "Name Server Design for Distributed Systems." in *Proceedings of the Seventh International Conference on Computer Communication*, October 30 to November 3, 1984. Sidney, Australia. Also available as USC/Information Sciences Institute. RS-84-132, June 1984.

25. Postel. J., and A. Westine, "Requests for Comments Summary." USC/Information Sciences Institute. RFC 899, May 1984.

26. Kirton, P., "Some OSI Experience of the ARPA-Internet," in *Proceedings of the Second International Conference on Introduction of Open Systems Interconnection Standards*. Ottawa, Canada, May 15, 1984. Also available as USC/Information Sciences Institute. RS-84-142, July 1984.

27. Katz, A., "An Experimental Internetwork Multimedia Mail System," in *Proceedings of the IFIP 6.5 Working Conference on Computer Message Services*, Nottingham, England, May 1984. Also available as USC/Information Sciences Institute, RS-84-134, June 1984.

28. Mockapetris, P.. "The Domain Name System." in *Proceedings of the IFIP 6.5 Working Conference on Computer Message Services*, Nottingham, England, May 1984. Also available as USC/Information Sciences Institute. RS-84-133, June 1984.

29. Reynolds, J., "ISI-Hosts," USC/Information Sciences Institute, June 1984.

30. Reynolds, J., and J. Postel, "Assigned Numbers." USC/Information Sciences Institute. RFC 900, June 1984.

31. Reynolds, J., and J. Postel, "Official ARPA-Internet Protocols." USC/Information Sciences Institute. RFC 901, June 1984.

32. DeSchon, A., *MCI Mail/ARPA Mail Forwarding*, USC/Information Sciences Institute. RR-84-141. August 1984.

9. PROJECT NEAT

Research Staff:
Danny Cohen
Larry Miller
Eric Anderson

Support Staff:
Jan Brooks

9.1 PROBLEM BEING SOLVED

As information-gathering tools become more and more powerful, aircraft pilots are faced with an ever-increasing amount of data, from which they must extract the information that is most relevant to their missions and most important for them to know at that moment. This "information overload" can far exceed the pilots' ability to comprehend the information, and may adversely affect their ability to react quickly and efficiently. What is needed is an intelligent machine copilot, or Pilot's Associate, which monitors the voluminous amount of flight data and presents the proper information to the pilot over multiple communications channels (including visual displays and voice). This system must be intelligent, flexible, and utterly dependable. The pilot should be able to tailor the displays according to his preferences, and to bring up information where and when he desires.

The NeaT project is studying the problem of information transfer from sophisticated aircraft systems to pilots. To accomplish this research task, the project will design an advanced, information-oriented man-machine interface for aircraft, which will behave as an intelligent associate to the pilot. The interface designed by the NeaT project will be applicable to a wide range of aircraft: military and civilian, light and heavy, combat and transport. The interface will be based on an expert system, which models the decisions of an experienced copilot. The system will be knowledgeable in areas such as fuel management, navigation, and aircraft reconfiguration in response to damage or emergency. A major focus of the project will be the interface between this expert system and the pilot: the project will study various methods and philosophies of information presentation, with an emphasis on intelligent speech and modifiable visual displays. Besides building a working generic system, this research will produce a body of knowledge that future researchers may build upon.

9.2 GOALS AND APPROACH

The NeaT project will create a generic cockpit simulator for use in developing and demonstrating the Pilot's Associate. The simulator will incorporate high-resolution computer-generated displays, multimodal man-computer communication channels, and interfaces to three computation facilities: a graphics processor, a general-purpose processor, and an artificial intelligence processor. The graphics processor will be the fastest and will deal with the lowest level of details, whereas the AI processor will deal with the highest level of knowledge and will probably be the slowest. The graphics and general-purpose processors will handle all dynamic simulation and most display issues. The AI machine will be an expert system, involved in global situation comprehension and in the generation of high-level communication with the pilot.

Using the generic aircraft simulator, the project will pursue four areas of investigation into man-machine interfaces:

1. **Information presentation.** The expert system will assess the available data and decide which information to present to the pilot in any given situation.

2. **Information delivery.** The project will develop a multimedia communications channel from the system to the pilot, including visual (schematic and pictorial displays) and audio (Intelligent Speech output).
3. **System modification language.** The project will design a display management language by which pilots can communicate with and modify the system.
4. **Confidence.** The project will investigate techniques to maximize pilots' confidence in the system.

9.2.1 Information Presentation

A popular approach to information presentation is to deliver to the pilot all available information at all times, so that he may have immediate access to any data. A major advantage of this approach is that it has higher throughput of information delivery and that it decouples data presentation from data consumption. A major disadvantage is that this higher throughput may lead to information overload, because the amount of data exceeds the pilots's ability to absorb and comprehend it.

This project will concentrate on another approach to information presentation, in which a subset of the total information is presented according to the pilot's needs and the priority and relevance of the data. The project will investigate the application of a rules-based, mathematical approach to information prioritization and delivery. The key issue is the methodology of deciding which information to deliver, when, and how. This decision should be based on the expert system's internal knowledge and on the pilot's requests, directives, and guidelines.

9.2.2 Information Delivery

The following modalities will be used for information delivery to the pilot:

- display of schematic forms (alphanumeric, conventional instruments, etc.),
- display of pictorial images,
- Intelligent Speech to the pilot, and
- various low-bandwidth audio and visual devices.

The schematic forms will start as familiar conventional instruments with added flexibility, such as the ability to display relevant markings computed dynamically, or to use variable scales (e.g., finer resolution for altitude display at lower levels). Under program control it will be possible to modify the information delivered by each instrument, as well as the way in which it is delivered. It will also be possible to combine instruments and to invent new ones.

In order to facilitate the design and versatility of cockpit displays, the project will use a "display builder" graphics editor. The display builder is a graphics front-end to the display tables, which also may be directly manipulated by the display designer or the pilot. Pictorial displays will be used to convey information about maneuvering, navigation, and aircraft systems.

Intelligent Speech (IS) is one of the most challenging tasks of information delivery. Previous attempts to create talking systems have proved inefficient and have alienated many pilots from the use of machine-produced speech. The IS system produced by this project will be part of the integrated information delivery system and will be modeled as an intelligent associate, able to assess information and relay it to the pilot much as a co-pilot would

Low-bandwidth audio and visual "idiot lights" are used very freely in modern aircraft. The use of these devices will be minimal in this project; they will be used mostly for interrupts and to attract the pilot's attention.

9.2.3 Modification Language

The pilot needs to communicate many types of information to the system, such as mission objectives, the proposed plan, data about the aircraft, and weather conditions. In addition to this passive kind of data, the pilot needs to communicate active entities such as procedures and rules. Pilots also have a variety of preferences regarding the style of an information delivery system.

In order to allow this type of communication from the pilot to the system, the project will develop a display management language that pilots may use in customizing the system. The language will be an extension of the display builder tools used by the initial designers. With this language, pilots will be able to control a variety of parameters and styles, and will also be able to modify the operational and evaluation procedures used by the system. Pilots will be able to call for the instruments of their preference to display any information. This will be done through techniques that allow the pilot to interface directly to the display builder tables so that he has the option to display information in a number of preset formats, to combine information in a single display, and to display specific computed information of his choosing. The selection/programming techniques will range from selecting among existing preprogrammed instruments (digital, analog, thermometers, tapes, hands, needles, compass-roses, etc.) to describing new instruments and the information to be displayed on them. In a similar manner, the pilot will be able to program the system to alert him when certain conditions hold.

The language developed by the project must have certain important features: it must be simple and natural enough for military pilots (who typically are not computer programmers) to use, while being rich enough to express the concepts and procedures that these pilots may wish to communicate to the system. Unlike traditional programming languages, this language will not be restricted to typed input, but will probably include dial settings, pointing, and various other media, including speech as it becomes practical. The pilot will communicate his intent via the construction and alteration of rules. The rule predicates will be taken from the aircraft/navigation domain, and the actions will be control activations, voice messages, and so on.

9.2.4 Confidence

If the Pilot's Associate is to succeed, the pilot must trust it to provide needed assistance in many touchy and unpredictable situations. Building this type of confidence is a major issue. It is not sufficient that the Pilot's Associate take the correct action and then inform the pilot of the action taken. It must present its activities and decisions in such a way that the pilot's confidence in the correctness of its actions is enhanced. Pilot confidence must be maintained and renewed every time the system is used; pilots are rigorously trained that any mechanical device or system can break down, and that no mechanical aid, however familiar, should be trusted without proof that it works correctly.

It is essential to the issue of confidence that the system work correctly and consistently. However, even a perfect system must convince its users of its reliability. Recent work [1] indicates that an expert system should be able to explain its actions to its human user. To this end, the project will build explanatory networks involving explanations of how decisions were reached. By explaining its actions, the system will demonstrate to its users its "reasoning power" and logic, which can be applied to other situations. Through explanations and actions taken (or not taken), the system must convince the pilot that it can be extended to handle unpredicted situations, or at least to detect when it cannot handle a situation and to ask for human assistance.

In general, the system will verify the consistency of its model against real-time readings to obtain a calibration of the situation, of the sensors, and of the model itself. Any deviations (excluding minor, predictable inconsistencies, which will be incorporated into the model) will be handled explicitly, including those that require human assistance to resolve. Advising the pilot about the correlation between various independently derived parameters may contribute significantly to the pilot's confidence level. For example, a notification such as "ETA is 57 minutes according to both the DME reading and our previous estimate" contributes to the pilot's confidence in both the DME and the estimation process.

9.3 SCIENTIFIC PROGRESS

A generic flight simulator has been built. It uses state-of-the-art graphics display hardware (SGI's IRIS display) and provides the initial testbed for information display to the pilot. The simulator provides real-time (simulated) aircraft control and conventional instrument displays. The software for the simulator is written in C and runs on a VAX/11-750. This software allows for straightforward alteration of display formats, both in the information to be displayed and in the way it is displayed.

An earlier version of the simulator has been used to investigate the Intelligent Speech model—what to say to the pilot and when to say it. The IS subsystem is an expert system with knowledge of specific information needs during several regimes, particularly in the critical low-visibility approach mode. It is based on a mathematical representation of event priorities and the pilot's span of attention.

The mission-oriented expert system design has been completed. Currently, four substantive areas of expert system research and development need to be accomplished:

1. Real-time decision-making.
2. Decision-making with unreliable, error-prone information, and potentially conflicting information sources.
3. Fail-safe decision-making. The pilot's associate must not make decisions or take actions that are potentially harmful to the mission.
4. Explanation of the expert's actions: the reasons for decisions taken and not taken.

The initial phase of the project has concentrated on the error/anomaly detection problem: what events and data to monitor; how to determine when conditions are out of normal range; how to determine whether the errors are real, or are faults of the instruments or the external world (a navigation station goes offline, for example).

The following items are currently under development:

- The pilot's language for expressing his intent.
- The interface between the graphics processor and the AI processor. Currently we are working on interfacing a Symbolics 3600 to the IRIS workstation.
- The expert system. A subset of the rules for flight are being developed. The rules for Intelligent Speech are complete. The rules for error detection are under development; ones for most normal flight regimes have been completed.

9.4 IMPACT

The major product of the NeaT project will be a highly capable generic aircraft information system. The results of our research will guide aircraft designers in the development of future cockpits, leading to a more efficient and reliable environment for pilots. The generic design of our Pilot's Associate will allow its principles to be applied to a variety of aircraft and mission types.

The NeaT project will also have benefits beyond this specific application. Our research will solve problems in connecting a diverse collection of hardware and software capabilities in order to support real-time knowledge processing tasks. In addition, we will build an expert system with a very high level of sophistication in its probability and environment models. Our research is an important step in the application of expert systems to a wide range of critical real-time environments.

9.5 FUTURE WORK

In the next year, the NeaT project will expand the computing environment in the cockpit simulator, specifically providing more advanced input and output capability. We will continue exploring visual and audio information presentation techniques. In addition, we will continue to develop the pilot's system modification language.

In our work to interface the graphics processor to the AI processor, we will complete the interface between the Symbolics 3600 and the IRIS workstation.

REFERENCES

1. Swartout, W., "XPLAIN: A system for creating and explaining expert consulting systems," *Artificial Intelligence* 21, (3), September 1983, 285–325. Also available as USC/Information Sciences Institute, RS-83-4.

10. STRATEGIC COMPUTING INFORMATION SYSTEM

***Research Staff*:**
Danny Cohen
Leroy Richardson

***Support Staff*:**
Jan Brooks

10.1 PROBLEM BEING SOLVED

The success of the Department of Defense's Strategic Computing research program depends on the easy dissemination of program-related information to a geographically distributed research community. The Strategic Computing Information System (SCIS) project provides an automatic and efficient system for information exchange between the Strategic Computing personnel and the research community-at-large.

10.2 GOALS AND APPROACH

ISI designed, implemented, and currently operates a similar information system: the MOS Implementation System (MOSIS—see Chapter 5). It uses an automated user interface which provides the information needed for successful interaction with MOSIS over the ARPANET or TeleMail. New users are routinely introduced to MOSIS via various electronic mail systems without any assistance from MOSIS personnel. MOSIS also handles various communication-related issues, such as message fragmentation, check-sums of submitted design files, acknowledgments, and duplicate detection.

Initially, a basic SCIS will be implemented on TOPS-20, using modified code originally developed for MOSIS. Once the SCIS is in operation, it will be extended to IBM-PC and VAX implementations. These later implementations of the SCIS are expect d to be more complicated, allowing us to incorporate the lessons we have learned from the first use on the TOPS-20 system.

The receipients will be those who can send and receive electronic mail to the SCIS. Initially, this group will include the ARPA-Internet mail community (ARPANET, MILNET, CSNET, DEC, Xerox, etc.) and the TeleMail users; soon, it will include MCI Mail (including TELEX) and TYMNET users.
Typical transactions of the Strategic Computing Information System will be:

- announcements;
- requests for information/information retrieval;
- Commerce Business Daily/Request for Proposal-related activities;
- information dissemination;
- queries;
- administrative requests (join, address change, etc.).

All such communication will be handled without exception via electronic mail (ARPANET, TeleMail, MCI Mail, TYMNET).

10.3 SCIENTIFIC PROGRESS

The first months of the SCIS project were devoted to the design of the system as it is outlined below. Implementation of the SCIS will begin in the next reporting period.

In October 1984, SCIS will begin posting all Strategic Computing-related Commerce Business Daily (CBD) announcements. Eventually, they will be divided into several groups and topics (each with its own index), such as RECENT, OPEN, AI. SUPPORT, VLSI, CMOS. GaAs. and PCB. Each document may appear in several indexes.

The basic system will support the following requests:

- INFORMATION request:
  - Retrieves documents from the pool of SCIS documents. A document is retrieved by giving its document identifier.
  - The document indentifier "ALL" retrieves the titles and indentifiers of all available documents.
- INDEX request:
  - Retrieves a list of titles and indentifiers of the documents available for a specific topic.
  - The topic "TOPICS" retrieves all summaries for which indexes are available.
- ATTENTION request:
  - The message is forwarded to a mailbox for manual handling by the SCIS staff.

Document attributes include an access code, ID, dates (of posting and revisions), owner (key), title, abstract, keywords, and references (companion/related documents). Documents may be structured to include related documents.

Information can be retrieved from direct information (title, ID, date, keywords) or from indirect information (table of contents, subject). The following is an example of a typical retrieval request:

```
  Retrieve:  Document/Name/Abstract
        of:  ID/F (keywords)
     since:  date
 ref-level:  n   (key)
```

Documents can be posted only by those with proper authorization and access control. Announcements can be made to bulletin boards or to explicit mailing lists. Bulletin boards will be implemented by the TOC mechanism and by the various mailing lists.

Mailing lists will have owners and keys. Individual users will be able to remove themselves from mailing lists, but will not be able to add themselves. Additions to mailing lists will be controlled by their owners.

The following is a list of typical messages to SCIS:

- I want to join.
- Please provide document # xxx, with references.

- Please provide a list of all documents/abstracts re: xxx.
- Please add me to/remove me from XXX-ML (a mailing list).
- My new address is xxx.
- Are you interested in my new invention/document?

The user interface design will be forgiving (DWIM: do-what-I-mean) and adaptive, and it will offer built-in help.

10.4 IMPACT

SCIS will function as a technological gatekeeper for Strategic Computing-related information. It will provide users with an efficient method for locating references and information, mostly using topic-specific indexes and abstracts.

Increased current awareness of the state of the art will effectively support the development of Strategic Computing technology.

10.5 FUTURE WORK

In addition to the forthcoming implementations described in the previous sections, we will begin to translate the MAINSAIL-based MOSIS code into the more portable C language; thus, we will be able to provide the SCIS to most UNIX systems as well as to personal computers.

11. WIDEBAND COMMUNICATIONS

Research Staff:	*Research Assistant*:	*Support Staff*:
Stephen Casner	Ehoud Haberman	Jan Brooks
Alfred Beebe		Jeff LaCoss
E. Randolph Cole		Jerry Wills
Ian Merritt		
David Walden		

11.1 INTRODUCTION

The Wideband Communications (WBC) project is one of several groups involved in the joint DARPA/DCA Wideband Packet Satellite Program. The objective of the Wideband Program is to explore the technical and economic feasibility of packet voice on a large scale, and to investigate other media, such as packet video.

ISI's role is to conduct experiments using the Wideband Network as it becomes available for regular, active use. ISI has been closely involved in efforts to make the network as reliable and useful as possible, and has established a schedule of regular tests and experiments in an effort to encourage real use of the network, thus gaining as much practical experience as possible.

11.2 PROBLEM BEING SOLVED

As the military is asked to do more and more with less manpower, communications becomes increasingly important. Reaction times must be fast, and many different kinds of data must be transmitted, correlated, and analyzed. This is very difficult, even impossible, to achieve when different networks are used for voice, data, facsimile, video, and other media. It is difficult enough to provide voice communications—let alone data and other media—in a tactical environment.

The DARPA Network Secure Communication (NSC) program took one of the first steps toward integrated military communications when it provided voice communication along with data on a packet-switched network. The NSC program showed very convincingly that packet voice was feasible. Moreover, a study by the Network Analysis Corporation [3] showed that a single packet network for both voice and data could support the DoD's communications needs much more efficiently than conventional alternatives.

However, the data rate of the network used (the ARPANET) was not high enough to support a meaningful volume of voice traffic. In addition, the cost of a packet voice terminal (speech coder and decoder and network interface) was prohibitively high.

Advances in VLSI have now made it possible to build a packet voice terminal, about the size of a normal telephone, for a few thousand dollars. However, a packet network with a data rate high enough to support reasonable volumes of voice traffic was still needed.

The DARPA/DCA Wideband Packet Satellite Network was built in 1980 to provide a high-rate packet network for experiments with multiple packet voice streams and other high-rate data. The

Wideband Network provides a raw data rate of about 3 Mb/s. which is more than a thousand times the data rate for a narrowband speech compression technique such as LPC.

The Wideband Network is currently being actively used for packet voice experiments. Since the packet voice terminals used have not yet been produced in large numbers. much of the load is generated by traffic simulators at various points in the network. The WBC project has expended considerable effort in developing an interface which provides voice access to the Wideband Network from ordinary telephones, in order to promote everyday use of the network and gain realistic user experience as early as possible.

Since the Wideband Network can transmit data at a rate more than fifty times that of the ARPANET, it will allow experiments with other media. such as packetized compressed video, which require inherently high data rates. Since video signals share many of the characteristics of voice signals (i.e., a "bursty" nature and a high degree of redundancy), the WBC project is conducting experiments to demonstrate the utility of packet video in the same way that the NSC program demonstrated the utility of packet voice. Thus the Wideband Network will serve as a research testbed for packet video, just as the ARPANET served as the original research testbed for packet voice.

The Wideband Network will help further extend the concepts of integrated networking to much higher data rates. while maintaining all the advantages of security and survivability which are inherent in packet switching. In addition, the Wideband Network is powerful enough to connect together high-bandwidth local area networks (LANs) for the first time.

Compared to existing terrestrial packet networks, satellite-based networks such as the Wideband Network have some disadvantages: longer delay and a higher bit error rate. However, satellite-based networks have a much higher typical data rate and an inherent broadcast nature. Communications systems using the Wideband Network will be designed to take advantage of its strong points and to compensate as much as possible for its weaknesses. Much of the research task will lie in designing the communications systems to fit the characteristics of the network.

11.3 GOALS AND APPROACH

The initial goal of the DARPA/DCA Wideband Program is to test and validate the feasibility of packet voice on a large scale. Later, attention will be focused in other directions, including experiments with high-rate transmission of more conventional non-real-time data and the application of packet-switching techniques to media such as packet video for the first time.

The overall approach of the ISI WBC project is twofold: to expedite the transition of mature technology, such as packet voice, from the laboratory into the everyday working environment: and to apply the unique capabilities of the Wideband Network to new areas, such as packet video and high-rate data transmission. This approach requires a broad spectrum of efforts, including development of hardware, software, and protocols. Development of research technology into practical systems requires a combination of highly reliable hardware and robust, easy-to-use software. Development in new areas such as packet video requires powerful new hardware which is flexible and programmable, along with development and testing of new software and protocols.

The ISI WBC project has been working in four specific areas:

1. distribution. support, and improvement of the Switched Telephone Network Interface (STNI). a device designed to greatly expand the availability of packet voice and encourage its regular use;
2. design and implementation of hardware and software for packet video experiments. including offline software simulations of the algorithms to be implemented:
3. installation of hardware at a second site (the MIT Lincoln Laboratory), which will participate with ISI in packet video experiments;
4. enhancement of Wideband Network reliability by testing network performance and working to solve system problems.

Useful experience with any new technology is the result of its exposure to a large number of potential users. Even though the cost and size of packet voice terminals have decreased greatly, as demonstrated by the Lincoln Laboratory PVT. existing terminals are still too large and expensive to supply to large communities of users who may use them only occasionally. To expand the availability of packet voice and gain the experience of a wider user community, the ISI WBC project designed and built the STNI [1] and distributed it to other Wideband sites.

The ISI WBC project is working to answer the question: "Is video as well suited to the packet-switched network environment as voice?" Video signals, like voice, are "bursty," i.e., the picture can change rapidly from one scene to the next and then stay fairly constant for some time. Current digital video transmission systems are designed to work with fixed-capacity networks, and they use the full channel capacity whether or not it is required. Packet-switched systems, on the other hand, are designed to carry bursty traffic, taking advantage of the fact that, given a large number of sources of traffic, the overall network load stays fairly constant.

The goal of the packet video experiments is to build a video transmission system which is optimized for a packet environment. This goal permits—even dictates—the design of a system which can detect change, or the lack of change, from frame to frame in the video input. Other characteristics of packet networks, such as selectable priority on a per-packet basis, can be used to allow the network to discard less important video information while making sure the most important video information always gets through.

It should be possible to build a system capable of transmitting television-type video with moderate motion at a data rate of 1.5 megabits per second. Our approach has been to survey the literature and technology of video bandwidth compression and select the best overall compression technique, and then to develop a hardware/software system that is as programmable and flexible as possible in order to support a variety of experiments.

Programmability makes the ISI system unique. Although commercial video bandwidth compression systems are available, they are not programmable in any way, and they cannot be customized to take advantage of the Wideband packet satellite channel.

In order to provide a realistic, useful test of the ISI packet video system, an identical set of hardware for packet video experiments is being assembled at Lincoln Laboratory, and video transmissions across the Wideband Network will begin as soon as the necessary hardware and software are available.

11.4 SCIENTIFIC PROGRESS

11.4.1 STNI—Switched Telephone Network Interface

In order to increase the availability of the Wideband Network for potential voice users. ISI has designed the Switched Telephone Network Interface (STNI) [1] The STNI sits between a telephone line and a Wideband Network voice terminal (the Lincoln Laboratory PVT), and receives commands from either DTMF (Touch Tone) signals from the user's telephone or digitally from the PVT. The STNI is based on a Z80 microprocessor, and consists of a single 7-inch by 7-inch circuit board which plugs directly into the Lincoln Laboratory PVT (or it can be used stand-alone).

To handle an incoming call, the STNI first answers the ring and sends the caller a second dial tone. The user then commands the interface with a series of key presses (called a dialing sequence), and the interface uses that information to make a Wideband Network voice connection, either to a normal PVT or to a PVT which contains another STNI.

To generate an outgoing call, the user first makes a Wideband Network voice connection to a remote PVT with an STNI. Connection information is passed over the Wideband Network to the remote PVT, which instructs its STNI to pick up the line and place a call.

STNIs are now permanently installed in PVTs at ISI, Defense Communications Engineering Center, Lincoln Laboratory, and SRI International. SRI modified two additonal STNIs for use as the standard input/output device for their Packet Radio Network speech terminals. The STNI-equipped PVTs are serving as the primary test vehicles for voice calls on the Wideband network.

A number of improvements have been made to the STNI software and to the related PVT software:

- the output volume level was increased to make the STNI useable over longer distance telephone connections;
- a special "access mode" with a higher pitched dialtone was added to allow commands to be entered through DTMF tones during a conversation; and
- table entries were added to provide automatic routing of calls to New England and Washington, D.C., area codes.

A pesky problem with echoes results when any device is connected to the switched telephone network. Echoes are not apparent when there is little or no delay in the audio path or when the communication is half duplex (one direction at a time). However, the Wideband packet voice network is full duplex, and the satellite link introduces a relatively long delay (230 milliseconds).

The current version of the STNI uses a technique called echo supression, which basically forces the voice conversation to be half duplex, i.e., one party cannot speak while speech is being received from the other party. A device called an echo canceller can eliminate the echo problem, but these devices have been relatively large and expensive. Newly introduced microprocesors capable of real-time signal processing can be programmed to do echo cancellation, and future plans include integrating a single-chip echo canceller into the STNI.

11.4.2 The Wideband Program Task Force

As is the case with any research and development tool, the amount of progress on Wideband Network applications depends largely on the availability of the Wideband Network itself. To ensure maximum availability, two kinds of effort are needed: a sustained effort to keep all the components of the system aligned, calibrated, and working properly, and an occasional concerted effort to solve a system problem or implement a new feature or function.

The Wideband Task Force was established in March 1983, with a participant from each of the primary Wideband Program contractors, including ISI. The job of the Task Force was to focus a large amount of effort and attention on a number of then-existing problems, and any others which might be uncovered in the course of achieving full operation.

The method chosen by the Task Force was to identify the problem areas, assemble the key people required to investigate and fix the problem at a single site, and let them work on the problem for a week, or as long as it took to identify and solve the problem. Generally, more than one problem was targeted.

During this reporting period, the Task Force convened for seven on-site work sessions to focus on specific problems. The following accomplishments resulted from the sessions and from individual efforts between sessions:

- several system problems were identified and isolated in both the Pluribus Satellite IMP (PSAT) and the Earth Station Interface (ESI), and most of these problems have been resolved;
- extensive and routine operation at the full network bit rate of 3.088 Mb/s was achieved;
- *seven sites were brought to operational status; and*
- interburst spacing was reduced by 50 percent for increased efficiency, made possible by improvements in the network subsystems.

11.4.3 Packet Video

The objective of the packet video work at ISI is to determine if there is a good match between the needs of a video transmission system and the capabilities of packet-switched networks, as was the case with voice. The Wideband Network is the first long-haul packet-switched network which can support such an experiment.

Digital transmission of video signals is nothing new. Neither are bandwidth compression systems which can reduce the data rate of a video signal with moderate motion from the normal 100 Mb/s to 1.5 Mb/s or even less. However, such systems are designed to work with fixed-rate channels, are hard-wired to perform only one set of functions, and are very expensive.

The packet video experiments are based on the following facts and observations:

- The required data rate for video communication is the product of the spatial (x and y), shade and color (z), and temporal (t) resolutions used.
- The scene being transmitted will remain fairly constant except for occasional sudden changes as the camera is zoomed, panned, or switched.
- The system must cope with sudden changes in channel performance (data rate, BER, delay, etc.).
- Both the video encoding and decoding equipment will have significant storage and processing ability.

Therefore, the packet video system must

- be as flexible and programmable as possible to handle new situations and to exploit as yet unknown opportunities;
- detect motion, or the lack of motion, and transmit as much or as little data as necessary;
- degrade gradually and gracefully, rather than suddenly and drastically, in response to decreasing channel capacity or increasing channel error rate.

These and related issues in satellite video communications are discussed in much greater detail in [4].

The ISI Wideband Communications project is engaged in designing and building a programmable system for packet video communications. Figure 11-1 is a block diagram of the ISI system. The hardware is based on a commercial video I/O/frame buffer system built by ADAGE, Inc., chosen because of its modular structure, which is based on a wide, fast, backplane bus. The ISI-built hardware consists of the Video Engine, a two-card set which performs the block transform processing in real time, and a high-speed serial interface for input and output of coded image data.

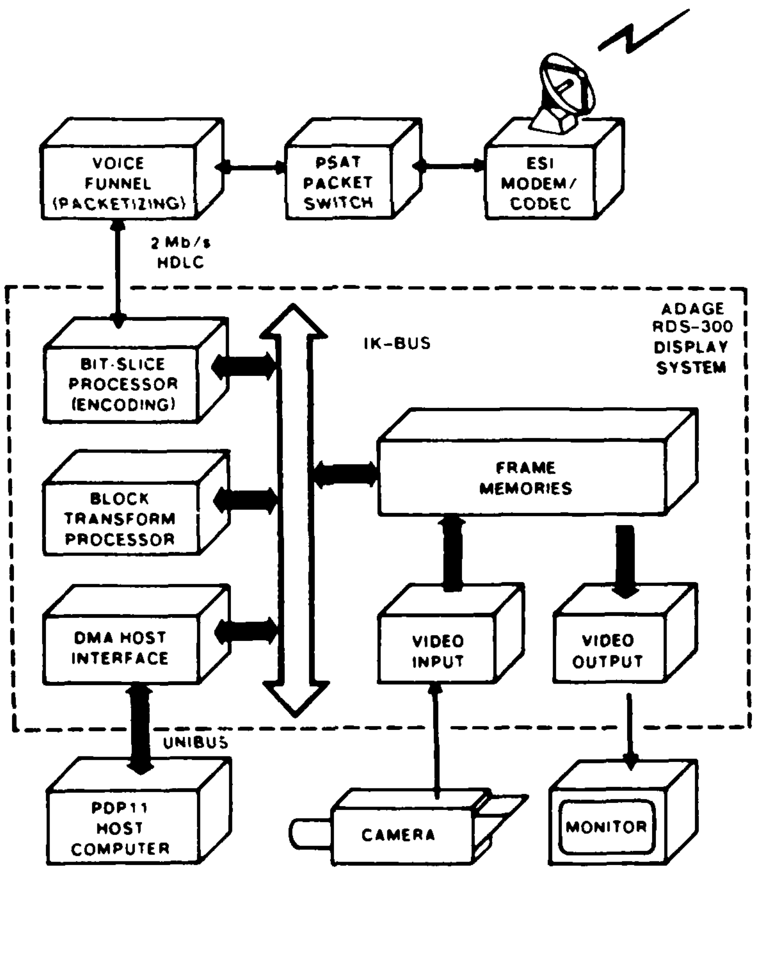


Figure 11-1: Block diagram of real-time video system

The following steps have been accomplished to date:

- A literature study was conducted and a candidate family of algorithms (block transform coding using the Discrete Cosine Transform, or DCT) was chosen.
- ADAGE raster graphics systems were acquired and installed at ISI and Lincoln Laboratory to provide the foundation on which to build the programmable video bandwidth compression hardware.
- The Packet Video Protocol (PVP) was designed and published [2].
- Software simulation of the hardware and the initial algorithms was completed.
- The hardware design was completed and the first Video Engine card set was constructed.

11.4.3.1 The Video Engine

The key feature of equipment for packet video experimentation must be *programmability*. Flexible, programmable hardware minimizes the number of unalterable decisions which have to be made and which may later prevent the system from taking advantage of new opportunities. In addition, programmable hardware allows a relatively simple system to be built first, with advanced features added later. Unfortunately, none of the commercially available hardware for real-time video processing is user-programmable.

Therefore, the WBC project chose to use a commercial product (the ADAGE RDS-3000 system) to handle video input and output and frame buffering, and to design and build additional hardware to do the real-time video bandwidth compression and high-speed serial I/O tasks. The additional ISI-built hardware is called the Video Engine.

The Video Engine is a single-instruction, multiple-data machine based on 8 Texas Instruments TMS 320 microprocessors. The TMS 320 is a 16-bit signal-processing-oriented microprocessor which executes one instruction every 200 nanoseconds, with a full 16x16 multiplier, a 32-bit accumulator, and a small (144-word) internal RAM. Thus the Video Engine operates at a rate of 40 MIPS.

The Video Engine is designed to be installed in the ADAGE RDS-3000 raster graphics system. The ADAGE system provides video input, frame buffering, and video output, and contains a high-speed 32-bit bus, which the Video Engine will use for its source and destination of video data. More than one Video Engine can be put on a single RDS-3000, but their computational throughput will ultimately be limited by the data rate.

The primary design goal of the Video Engine was to achieve maximum programmability and flexibility at a reasonable cost. The conventional architectures for abundant processing power for video applications are based either on bit-slice microprocessors or on dedicated, special-purpose logic.

Since the Video Engine has to be programmable, a dedicated, special-purpose (i.e., inflexible) processor was not considered. Bit-slice microprocessors can be used to achieve considerable computing power at a fairly reasonable cost, but they are somewhat difficult to program and can achieve at best about 5 to 10 MIPS per processor if flexibility is to be retained.

Microprocessors designed for signal processing, such as the TMS 320 and the NEC 7720, are fairly new. The cost per unit is fairly high, but it can be expected to decrease rapidly. The 320 and the 7720 are easier to program than a bit-slice microprocessor, though harder than an ordinary micro. Their architecture is fairly primitive compared to that of a conventional micro, but they are more than an order of magnitude faster.

Since algorithms for video and other real-time signal processing are generally parallel in nature, a parallel architecture with many identical processors is a good fit to the task. In addition, such an architecture is much more suitable for eventual VLSI implementation. We chose the multi-microprocessor architecture for these reasons.

We chose the TMS 320 over the NEC 7720, because it can operate from a fairly large (4K) external program RAM, compared to the 512-word internal ROM or EPROM used by the 7720. It is slightly faster than the 7720 for transform-oriented programs.

Figure 11-2 is a block diagram of the Video Engine, which has two main components:

1. a RAM-based sequencer to control data flow and supervise the operation of the 320s;
2. the eight 320s themselves, along with an external buffer memory and an external table memory for each, and a single external program memory.

The sequencer uses a 4K by 40-bit RAM for control, and operates with a 100ns cycle time. In addition to controlling the 320s, the sequencer is responsible for reading and writing pixel data via the IK-bus. The 40-bit sequencer microcode word is horizontal, i.e., the bits in the word directly control the hardware and the fields are not decoded. A single-level subroutine capability is provided.

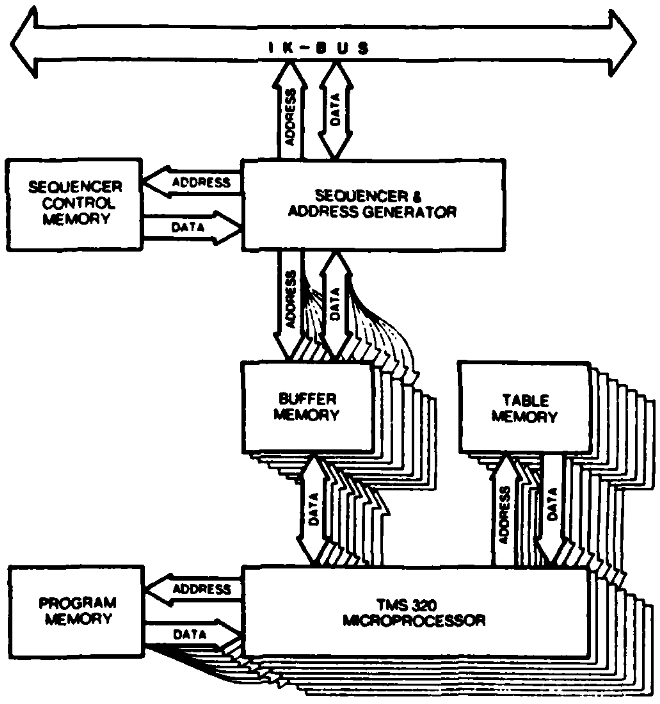


Figure 11-2: Block diagram of Video Engine

The Video Engine is designed to operate on blocks of data, typically 8 by 8, 8 by 16 or 16 by 8, 16 by 16, etc. A typical application would allocate a different block to each of the eight 320s. Blocks can be overlapped, but the RDS-3000 is structured to read groups of 8 pixels in a row most efficiently, and therefore blocks always start on 8-pixel boundaries horizontally. No such limitation exists vertically.

The sequencer addresses a pixel by taking a frame start address, adding a block offset to it, and adding a pixel offset to the result. The block and pixel offsets are fairly large, so a block could consist, for example, of every nth group of 8 pixels horizontally and/or every mth pixel vertically. The block offset is larger than a frame, allowing for even more flexibility.

Each 320 has a buffer RAM of 4K 16-bit words associated with it. The buffer RAM smooths out the bursty nature of the processing, allowing the 320s to run at full speed, and augments the small RAM in the 320s. Buffer RAM contents are transferred to and from the bulk frame memory under control of the sequencer, and transferred to and from the 320s via their I/O ports. Each 320 also has a table RAM (4K 8-bit words) which can be used for storage of coding and decoding tables, etc. This allows each 320 to code and decode different data, even though all the 320s are executing the exact program code simultaneously.

Only one of the 320s has a program memory associated with it; the other seven 320s are synchronized to that one. A simple memory mapping mechanism allows the use of 16K words of program memory, compared to the normal 4K word maximum. This mechanism allows code for several algorithms to reside in memory at once, perhaps allowing the use of different algorithms for different channel error rates, etc.

The Video Engine consists of a set of two cards, with the sequencer on one card and the 320s and their associated memories on the other. Power consumption is in the 100-watt range, but the RDS-3000 is able to supply adequate power and cooling. Logic is Schottky and low-power Schottky. The memories are 4K by 4 high-speed (55ns) static RAMS.

To help in hardware and software testing, all of the memories in the Video Engine can be loaded and read from the IK-bus. The RDS-3000 system is installed on a PDP 11/45, and the 11/45 or other host is responsibile for loading and starting the Video Engine.

11.4.3.2 Hardware and algorithm simulation

To support the packet video effort, it is necessary to simulate (1) the video bandwidth compression algorithms and (2) the video processing hardware. Both elements are integrated in an accurate simulation of the complete video processing system. The simulation package, called DCTSIM, pays particular attention to the simulation of interfaces between the components of the system.

Simulation of the video bandwidth compression currently implements the initial algorithm to be used in the real-time system, and transforms and encodes blocks of 8 by 8 or 16 by 16 pixels. The resulting compressed image has between one and two bits per pixel. When images coded to two bits per pixel are decoded and inverse transformed, the resulting image is virtually identical to the original. Coding to one bit per pixel causes noticable degradations in the reconstituted image, with quality increasing with the number of bits per pixel. The algorithm is currently being "tuned" to achieve a compression to somewhere between one and two bits per pixel without noticable degradation.

The simulation provides an important baseline for implementing the real-time version. Using a static image, it is possible to compare the results of the two implementations bit-for-bit to verify that the real-time version operates correctly.

The other function of DCTSIM is to test and develop the sequencer portion of the video processing hardware. The simulator is functionally equivalent to the hardware, and the code that drives the simulator is exactly the same as that which will drive the real-time hardware. The simulator has already proved very useful in assessing the functionality of the real-time hardware and pointing out errors and omissions in the design.

A new feature was added to produce timing diagrams of the simulated hardware signals. These diagrams can be used to document the hardware operation and to analyze timing and system bus loading, which would be hard to visualize in real time.

To allow the simulation of bandwidth compression on sequences of images, we have developed a program for the RDS-3000 which collects sequences of 12 video frames at the rate of 15 frames per second or slower. The frames of 256 by 256 pixels are stored in the frame buffer memory of the RDS-3000 and can be read out for storage on disks.

11.4.3.3 Real-time system implementation

ADAGE raster graphics systems are in place at ISI and at Lincoln Laboratory. These two sites will serve as the testbed for packet video experiments across the Wideband Network. Construction of the first Video Engine card set has been completed, and construction of the second set, to be installed at Lincoln Laboratory, is underway.

Both RDS-3000 systems have been equipped with a bit-slice processor (called a BPS), which is a high-speed microprogrammed machine designed to handle internal processing chores for an RDS-3000 system. Although the BPSs are too slow to perform the block transform function, they provide additional processing power for control of the entire real-time system, and they supervise the input and output of coded image data.

The Video Engine can now execute complete forward and inverse transforms of continuous video images in real time. This milestone includes

- the implementation of both the forward and the inverse Discrete Cosine Transforms in the TMS 320 microprocessor;
- microcode for the data-flow sequencer to process complete image frames; and
- software for the BPS processor to manage video image frame input and display.

11.5 IMPACT

DARPA-sponsored work continues to have a major impact in the packet voice area. Most high-speed local area networks (LANs) have incorporated packet voice, although most use it to store voice messages rather than for interactive communication. The advent of high-bandwidth LANs and powerful individual workstations has generated new requirements for high-bandwidth long-distance communication:

- to provide the same close-knit interoperation of workstations being developed on LANs for groups which are geographically distributed; and
- to serve as a resource-sharing mechanism for new generations of supercomputers, where users would prepare data using their workstations, transmit the data to the supercomputer for processing, and then receive the results on their workstations.

Sharing supercomputer resources is important both for the DARPA Strategic Computing program and for facilities being planned by NSF. The high-bandwidth packet technology being developed in the Wideband Satellite program has been proposed for both of these applications.

The Wideband Network is a jointly sponsored effort of DARPA and DCA. In addition to the original DCA Wideband Network site at DCEC, three additional sites (Fort Monmouth, New Jersey, Fort Huachuca, Arizona, and the Rome Air Development Center in Rome, New York) were brought into operation during this reporting period. The new sites will be an important mechanism for transfer of packet voice technology to the services. Personnel from one of the new sites have consulted with ISI WBC project members about providing video capability at some of the new sites. The ISI Packet Video Protocol (PVP) [2] could be used to support such an effort.

An architecture like that of the Video Engine should be well suited to any processing task that can take advantage of parallelism but does not require a large amount of code. Signal processing applications in general fall into this category.

The TMS 320, in particular, has had a big impact on the market, and improved versions of the 320 by Texas Instruments and similar chips by other manufacturers are certain to follow. The Video Engine is perhaps the first processor that attempts to use this type of chip in a highly parallel form. Such an architecture is likely to have a significant impact in the future.

11.6 FUTURE WORK

11.6.1 Packet Voice Experiments

Experiments with packet voice on the Wideband Network will continue over the next year or two, with new features and capabilities added, tested, and put into everyday use. ISI will continue to participate in the further development of Wideband packet voice, and will also continue to participate in efforts such the Task Force to upgrade the overall reliability and utility of the network itself.

The STNI is now serving as the primary test vehicle for packet voice on the Wideband Network, making continued support and improvement an important ISI task. To increase the quality and utility of the STNIs, echo cancellers will be built and retrofitted to the STNIs.

11.6.2 Packet Video

The immediate goal of the packet video effort is to get the initial real-time implementation up and running. Since the approach of the packet video effort is to start with a basic capability and steadily refine it, future packet video work will consist of increasing the performance of the video system by finding and exploiting ways to use the unique characteristics of the Wideband Network.

Basic intra-frame coding using the Discrete Cosine Transform is operational. We expect to improve the quality of this coding by tuning parameters; however, most of the reduction in required data rate will result from future work on inter-frame coding. By varying the frame rate to match the requirements of the image, we will be able to reduce the data rate when the scene is static. Alternatively, the intra-fame coding could be adjusted to produce higher quality (more bits) at the lower frame rate and remain within the overall data rate limitation. The adaptability of the programmable video bandwidth compression system and the packet technology of the Wideband Network are the key to future work in improving the performance of the video system.

REFERENCES

1. WBC Project, "Wideband Communications." in *1983 Annual Technical Report*, USC/Information Sciences Institute, SR–83–23, 1983.

2. Cole, E. R., PVP – A Packet Video Protocol. USC/Information Sciences Institute, W–Note 28. August 1981.

3. Gitman, I., and H. Frank. "Economic analysis of integrated voice and data networks: A case study," *Proceedings of the IEEE* 66, (11), November 1978. 1549–1570.

4. Casner, S. L., D. Cohen, and E. R. Cole, "Issues in satellite packet video communication." in *Conference Record of the IEEE International Conference on Communications*, IEEE. Boston, 1983. Also available as USC/Information Sciences Institute, RS–83–5.

WORKSTATION DIVISION

The Workstation Division concentrates on technology meant to support individuals whose productivity can be greatly enhanced with workstation-based computer tools. Since the typical information worker is computer-naive, the usefulness of computer-based tools is dependent on the ability of people to communicate with computers in a natural, free manner. Computer tools must be understandable, consistent, and easily directed; their development is the goal of the Workstation Division projects.

Knowledge Delivery Research provides technology allowing computers to express what they know in appropriate English. In many cases, this means the production of multiparagraph text that is easily read. Difficult theoretical problems in the production of such text include understanding the process of text generation; vocabulary and grammar characterization; and selection criteria for grammatical constructs, groupings, and orderings.

The Consistent User Environment (CUE) project provided an architecture in which software tools and users may interact with one another in a semantically consistent manner. Semantic consistency was assured through the use of detailed models of the expectations of users and the capabilities and requirements of the tools. The CUE system featured a menu- and forms-based user interface that reflected the underlying consistency of interaction. CUE addressed the issues of consistency in the user interface, building models of computer software and user interfaces and specifying interfaces between independently developed software modules. The user interface technology under development by CUE has been transferred to the Formalized System Development project for further exploration.

The Consul project explored the use of knowledge-based technology for providing a natural language interface between users and computer-based tools. The research concentrated on the use of detailed semantic models and inference mechanisms to reconcile user statements, user needs, and system requirements. The Consul approach was founded on the principle that the machine must have knowledge of what the user is trying to do with a program and what the program can do for the user. The theoretical issues addressed by Consul included the development of modeling techniques for very large models; natural language understanding and explanations; and knowledge acquisition. The Consul modeling technology has been transferred to a new project on Modeling Technology.

The following chapters discuss these projects in more detail, showing the goals, approaches, progress, and impact of each.

12. CONSUL

***Research Staff*:**	***Research Assistants*:**	***Support Staff*:**
William Mark	Gabriel Robins	Kathie Patten
Thomas Lipkis	Richard Stokey	
Norman Sondheimer		
Don Voreck		
Ralph Weischedel		
David Wilczynski		

12.1 PROBLEM BEING SOLVED

It is still very difficult for the average user to get a computer to do what he wants. If he has to combine more than one program to do the job, the task is even more difficult. Current programming technology provides little help in user interface design and implementation; the result is software that is very limited in terms of usability, adaptability, and uniformity across applications. The only exceptions are a few specialized packages whose user interfaces are carefully crafted for small sets of applications (at the cost of scores of man-years).

Even with the most careful crafting, these systems are inadequate for the needs of computer-naive users, who simply may not have the technological sophistication to use conventionally designed programs. Often, it is infeasible to train them.

Moreover, current programming/user interface technology does not project well into the future: it will be extremely difficult to extend systems built with current techniques to take advantage of advances in knowledge-based processing technology (e.g., speech, expert systems).

In summary, we believe that present technology cannot meet the increasing demands on user interface design, and is not well suited to the probable future character of interactive systems.

12.2 GOALS AND APPROACH

The Consul project explores the use of knowledge-based technology for providing a natural interface between users and computer programs. The research focuses on building detailed models of users and programs and linking them together through inference mechanisms. Actual communication between users and programs is in terms of a forms/menu interface, natural language requests, and generated natural language explanations.

Our approach is to offer a user-interface-based programming technology founded on the principle that the machine must have knowledge of what the user is trying to do with the program, as well as what the program can do for the user. For example, the machine must contain explicit knowledge of real world objects like spreadsheets, documents, and inventory—how they are implemented and how they are used. In this way, program design can be linked to user interface design so that the user can interact with the program in ways he thinks make sense, not just according to the way functions were "wired in" to the interface by the programmers. The use of explicit knowledge also allows the user to make requests and receive help in *his* terms, not just "sugared" versions of the program's internal format. Consul's knowledge base can also provide access to inferential routines that allow the system to react usefully to requests and commands that are ill-formed from the standpoint of the program's functional requirements. Finally, the knowledge base provides a foundation for incorporating software into future systems that include speech understanding and expert system technology.

The knowledge base is linked to real programs and data through an *acquisition dialogue* with the programmer. In this exchange, Consul uses inference to determine how the functions and data structures of the program fit into what the knowledge base already knows about the user and system environment. The software that is incorporated into Consul must be constructed in a manner that supports the flexibility and consistency of the user interface. That is, the programs and data structures must be written so that functions can potentially interact in any combination that makes sense semantically. This requires careful attention to software engineering principles during the programming process. Consul's interface is therefore designed to be used in conjunction with the CUE system (see Chapter 13), an environment that enforces the software engineering principles necessary to produce functions and data structures whose interactions can be semantically defined.

The combined Consul/CUE system runs in a distributed environment, with Consul residing on a server machine that can be shared by one or more CUE workstations. The idea is to achieve acceptable total system price/performance by distributing knowledge base and interface functionality across machine boundaries. By concentrating the high frequency window-oriented, menu-oriented, and command-oriented operations on the workstation, by putting the knowledge base on a server, and by designing the system to require only low-bandwidth communications with the server, we have a total system that is efficient enough for real-time use with currently available technology.

12.2.1 Knowledge Representation and Inference

Consul's knowledge base consists of a ***user model***—a coherent description of a "user's view" of the system; and the ***systems model***—a representation of the operations and data structures found in interactive systems. These distinctions are important for the inferential processes that use the knowledge base. For example, the user model provides the target for the natural language understanding system (anything the user says must be translated into some user model concepts) and the filter for producing explanations (anything said to the user must be in terms of user model concepts). The systems model similarly provides a filter for the acquisition process, allowing it to integrate only consistent application-specific descriptions into the knowledge base.

Consul's reasoning activity is based on the highly efficient ***classifier*** that we built as part of the NIKL system [8].[1] The classifier determines how each piece of incoming information fits in with the system's current knowledge of domain concepts. The task is to find the relationship between terms in the new information and the terms already known to Consul. For example, new information like "the meeting I just scheduled with Post" must be related to known terms like "schedule," "meeting," and "user."

Using these known terms, the ***realizer*** (the dual of the classifier [6]) associates the new information with the real data items it describes. For example, the realizer would associate the concepts representing "the meeting I just scheduled with Post" with a particular entry in the MEETING-FILE of

[1]NIKL [7] is a vastly improved reimplementation of the original KL-ONE system [1] that we have worked on cooperatively with Bolt Beranek and Newman, Inc.

a particular user. This enables Consul to process the right data based on user requests that contain only descriptions, not actual names of or pointers to data. The realizer also enables Consul to recognize the same piece of data if it is described in different ways (e.g., "the meeting I just scheduled with Post" could also be described as "my 1:30 meeting").

Once new information is recognized by establishing its relationship with known terms and data, it must be seen as relevant to the system's problem-solving goals. Consul does this by inference rules that state how the domain concepts represented in the knowledge base relate to the system's problem-solving methods (i.e., descriptions of programs to send mail, or display meetings).

In the Consul knowledge base, rules are special assertions relating two concept structures (a condition and a conclusion) in terms of correspondences between parts of the concepts. When incoming information instantiates the condition of a rule (as determined by the classifier), the inference mechanism checks the rule's assertional correspondences to determine if the new information can be reformulated according to the rule's conclusion concept. The correspondences express meta-knowledge about when it is appropriate (from the point of view of the system's problem-solving behavior) to reformulate information, e.g., "if you're trying to send a piece of text, check to see if it includes information about who is to receive it; if so, it can be redescribed as a message." When the inference mechanism checks the rule's assertional correspondences, it usually creates subproblems—requiring other rules to be applied—until the initial rule is satisfied and Consul knows how to respond to the incoming information via function execution or explanation.

12.2.2 Natural Language

Consul has demonstrated a workstation environment that integrates natural language, forms, menus, and commands into a single user interface [3]. Our research has focused on the natural language aspects of this interface; the rest of the interface research is part of the CUE Project—see Chapter 13. (Natural language understanding capabilities allow the user to state tasks even when he is unfamiliar with the details of a particular application, give him a way to state tasks that require the combination of two or more applications, and provide him with help and explanation in response to specific questions.)

The natural language understanding system uses a new semantic interpretation system, "Beaver," which is designed to efficiently compare parsed user input against a taxonomy of knowledge about English phrase structures [9, 10]. The result is a match with the concept that indicates the most specific interpretation of that part of the sentence. This interpretation is in turn tied through a rule-like assertional structure to a concept in Consul's user's model [11]. The English phrase structure taxonomy is implemented in NIKL and is thus an integral part of Consul's overall knowledge base.

12.2.3 Acquiring Services

Execution and explanation of individual application programs require detailed models of those programs in terms of Consul's knowledge base. We do not expect the programmer to build these models, since he is unfamiliar with the knowledge base and the modelling process. It is therefore essential that Consul provide computer aids for acquiring detailed models of application software from the programmer.

Consul's computer-aided acquisition mechanism consists of a structured "dialogue" between the system and the programmer. The dialogue is in terms of menu selection, form-filling, and generated natural language. The programmer presents the system with forms representing his programs and data structures. The system responds with forms representing the concepts in its knowledge base that are relevant to those programs and data structures. The programmer then (via indicating corresponding parts of the forms with the mouse) establishes links between his programs and data and Consul's knowledge base descriptions of the domain concepts those programs and data are intended to implement.

For example. the programmer may present a data structure representing electronic mail. The system presents him with a form displaying its "message" concept. The programmer links the concept piece by piece to his data structure, e.g., by specifying that the "subject" aspect of the message concept corresponds to the third field of the header block of his mail data structure. and so on. At any time. the programmer can indicate (with the mouse) that he wishes to have some aspect of the system's presentation explained to him. The system will then generate an English explanation of the concept selected, using the structure of the knowledge base, some English text generation rules, and a lexicon (see Figure 12-2). The programmer can examine this explanation and decide if he wants to make the link. In general, he will explore several different concepts in the knowledge base before making his choice. The result of this exploration and linkage process is a relationship between Consul's general model of a given function or data structure and its implementation in the application program. An example of the result of an acquisition dialogue is shown in Figure 12-1.

12.3 TECHNICAL ISSUES

The major technical problems of the Consul research effort are:

- *deciding what to put in the knowledge base and what to say about it*: We now have considerable experience with this process. and are using NIKL effectively to express our decisions.
- *developing a consistent formalism that allows automatic recognition and rule-based inference*: Our work. in combination with that of our colleagues. has been directed toward designing formalisms that preserve the epistemological structure inherent in the knowledge we are trying to represent. This work resulted in NIKL. and will hopefully continue to produce improved knowledge representation systems.
- *developing efficient algorithms for classification, realization, rule application, and semantic interpretation*: Consul's inference mechanisms need to take into account advances in our knowledge representation capability and the future availability of parallel computation.
- *providing mechanisms for interfacing the knowledge base to databases*: Explicitly modelling information in the knowledge base increases the system's capability. but is costly in terms of execution speed and memory requirements. Representing information in external databases can be more efficient (and is in fact necessary in some environments), but brings up issues of what the system does and does not know about the information it contains.
- *providing tools for creating, examining, and changing knowledge structures*: We have produced useful tools for each of these activities, but still have much to do in order to make our entry and maintenance procedures smooth and efficient.
- *using the knowledge base as a vehicle for expressing principles of user interface design, and enforcing those principles when the application software is built*: We have encoded a few such principles in the Consul knowledge base. These principles must be elaborated and tested with real users.

12.4 SCIENTIFIC PROGRESS

We have completed our work on the new implementation of KL-ONE (NIKL), in a joint effort with Bolt Beranek and Newman, Inc. We have already incorporated it into demonstration systems, and plan to use it in all of our future work. The classifier runs *50 times faster* in NIKL than in KL-ONE. In addition, NIKL provides more consistent and compact data structures for representing knowledge, along with a much-needed overhaul of the internal function-call interface.

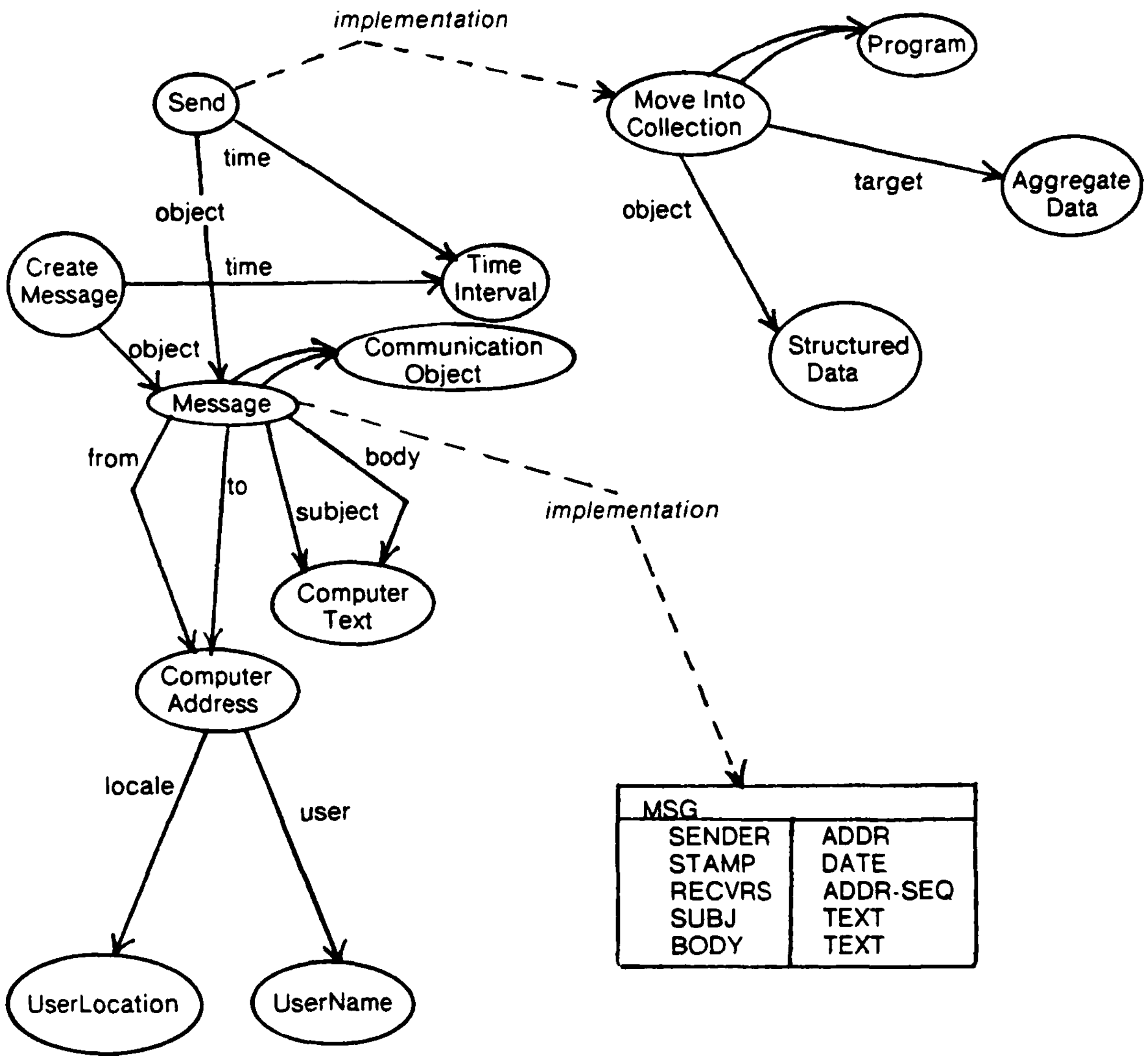
We have implemented a demonstration of our acquisition mechanism using NIKL and the CUE forms interface [4]. Figure 12-1 shows an example of the kind of acquisition this demonstration can perform. The data structure MSG from an electronic mail application is to be acquired. The figure assumes that it is already known that in this application, the user model concept "send" is implemented as a system model "move into collection" program. The programmer must show Consul which particular "structured data" is to be moved into the collection to implement "send". According to the model of "send", this structured data must correspond to the "message" concept. Therefore, in order to be consistent with what is known so far, the structured data—in this case MSG—must be seen as the implementation of the "message" concept. As shown in the figure, the result of the acquisition dialogue (about ten minutes' worth of exploration and form-filling) is the linkage between MSG and the "message" concept, and the final agreement by both Consul and the programmer that MSG is the implementation of "message".

We have implemented tools for browsing and explaining the concepts in the knowledge base. An example of the actual explanations generated for the concepts "message", "appointment record", "send" and "reschedule" is shown in Figure 12-2. We are continuing our cooperation in basic knowledge representation research with groups at Fairchild Research, Xerox Palo Alto Research Center, IntelliCorp, and Bolt Beranek and Newman, Inc. We have designed, and have begun the implementation of, a major new semantic interpretation system (Beaver) for natural language understanding in the NIKL environment [9, 10, 11].

We are continuing our investigation of parallel control structures for our knowledge base algorithms, in the context of MIT's Connection Machine architecture [12] and Holland's "Classifier System" [2].

The Consul project has been active with publications and presentations, including the following:

- Sponsoring, organizing, and conducting the 1983 Knowledge Representation Workshop: a gathering of the major researchers and implementers to discuss recent progress in knowledge representation research.
- Presenting a paper about knowledge-based user interface design at the Human Factors in Computing Systems Conference in Boston, Massachusetts [3].
- Publishing a paper about understanding ill-formed English input in the *American Journal of Computational Linguistics* [13].
- Chairing a session on "Intelligent Aids to Document Preparation" at the 1984 National Computer Conference.
- Presenting a paper about the Consul classifier at the International Joint Conference on Artificial Intelligence [8].
- Presenting a paper about combined natural language/graphics user interfaces at the 1984 Conference on Intelligent Systems and Machines [5].
- Presenting a paper about the natural language semantic interpretation system at the 10th International Conference on Computational Linguistics [10].



MSG	
SENDER	ADDR
STAMP	DATE
RECVRS	ADDR-SEQ
SUBJ	TEXT
BODY	TEXT

Figure 12-1: Result of Consul acquisition

For the concept "Message":

A ***message*** is a ***communication object*** that may have a sequence number, a location and a body. It must be "to" an ***active entity*** and it must be "from" an ***active entity***. A ***message*** is also ***structured information*** whose parts include several kinds of status, a subject, to specifications and a from specification. A ***message*** can be ***forwarded*** and ***answered***. Also, as ***structured information***, a ***message*** can be ***sent***, ***received***, ***deleted***, ***removed***, ***displayed***, ***created***, ***copied***, ***changed***.

For the concept "Appointment Record":

An ***appointment record*** is ***structured information*** that must describe an ***appointment***. Its parts include reasons, a meeting place, a time, and makers. An ***appointment record*** is also a ***timed computer object*** whose parts must be ***information***. As ***structured information***, an ***appointment record***, ***deleted***, ***removed***, ***displayed***, ***created***, ***copied***, ***changed***. As a ***timed computer object***, an ***appointment record*** can be ***scheduled*** and ***rescheduled***.

For the concept "Send":

Send has an object which must be ***information*** and a recipient which must be an ***active entity***. The ***information*** must be "to" a ***computer address***. When ***send*** is finished, the ***information*** is at the ***computer address***.

For the concept "Reschedule":

Reschedule has an object which must be a ***computer object***, an object-to-be-changed which must be a ***timed computer object***, a change specification which must be an ***update***, a time of occurrence which must be a ***time interval***, and an agent which must be a ***computer***. The ***time interval*** is before the ***update***'s ***time interval***. The ***timed computer object*** is transformed into the ***computer object*** according to the ***update***." When ***reschedule*** is finished, the ***computer object*** is accessible to the ***computer***.

Figure 12-2: Examples of Consul–generated English explanations

12.5 IMPACT

Consul research represents a fresh examination of the idea of a "personal support environment"—a computerized working environment for an individual that provides a significant resource for helping him do his job. Personal support environments open computers to a much wider audience, eventually to anyone with a need for their services. The natural interface accepts requests from users with no previous training with the system. It can explain how to use the system in terms that the user can understand and learn from. The knowledge base provides a basis for extension of the software into the realm of speech understanding and expert system technology. Natural interaction—especially speech—and knowledge-based expertise will be crucial for acceptance by users in many aspects of society where the need for computer service is greatest. The technology of knowledge-based personal support environments is fundamental to continued progress in the delivery of computer services to society at large.

12.6 FUTURE WORK

Our proposed effort in the area of personal support environments will concentrate in the following areas:

- ***Knowledge-based user interface design***: The environment must be easy for users to understand, ask questions about (in English), and use (via freely mixed English and graphics). This requires the system's user interface and the programmers' design framework to be based on the same internal knowledge base of assumptions and principles about system capabilities and usage patterns.
- ***Application acquisition***: New applications must be brought into the system, and existing applications must be modified whenever necessary without altering the basic model of how users use the system. This requires an acquisition procedure to unify all new software into overall system functionality based on the assumptions in the knowledge base.
- ***Distributed knowledge base architecture***: The system must be able to take advantage of a network environment consisting of a variety of machines. This will require research into how to split knowledge base and application software among the different processors in order to create an efficient execution environment.

Our goal is to deliver a working system to a user community for beta-test in two years, with full delivery in three years. We will focus on the computer-naive user within the military environment.

REFERENCES

1. Brachman, R., *A Structural Paradigm for Representing Knowledge*, Bolt Beranek and Newman, Inc., BBN Technical Report, 1978.

2. Forrest, S., *A Parallel Classifier for KL-ONE*, University of Michigan, Technical Report, 1984.

3. Kaczmarek, T., W. Mark, and N. Sondheimer, "The Consul/CUE interface: An integrated interactive environment," in *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 98-102, ACM/SIGCHI, December 1983.

4. Kaczmarek, T., "CUE Forms Description," CUE Working Paper, 1984.

5. Kaczmarek, T., and N. Sondheimer, "Man-machine interface modes: Variety and integration," in *Proceeedings of the 1984 Conference on Intelligent Systems and Machines*, Rochester, Michigan, April 1984.

6. Mark, W., "Realization," in J. G. Schmolze and R. J. Brachman (eds.). *Proceedings of the 1981 KL-ONE Workshop*. pp. 76–87. May 1982.

7. Moser. M. G., "An overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding*. Bolt Beranek and Newman, Inc., Cambridge. MA, 1983. BBN Technical Report 5421.

8. Schmolze, J., and T. Lipkis, "Classification in the KL-ONE knowledge representation system," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

9. Sondheimer, N. K., and R. M. Weischedel, *Consul Note 22: "Towards Semantic Processing with Phrasal Frames Using Structured Inheritance Networks"*. USC/Information Sciences Institute. 1984.

10. Sondheimer, N. K., R. M. Weischedel, and R. J. Bobrow, "Semantic interpretation using KL-ONE," in *Proceedings of Coling84*, pp. 101-107, Association for Computational Linguistics. July 1984.

11. Sondheimer, N. K., *Consul Note 24: "Translating to User Model"*, USC/Information Sciences Institute. 1984.

12. Stokey, R., *Consul Note 21: "Implementation of a KL-ONE Network on the Connection Machine"*, USC/Information Sciences Institute, 1983.

13. Weischedel, R. M., and N. K. Sondheimer, "Meta-rules as a basis for processing ill-formed input," *American Journal of Computational Linguistics* 9, (3–4), 1983. Also appears as USC/Information Sciences Institute, RS–84–146.

13. CUE

Research Staff:	Research Assistants:	Support Staff:
Thomas S. Kaczmarek	Nitsan Har-Gil	Kathie Patten
	Haym Hirsch	
	Gabriel Robins	

13.1 PROBLEM BEING SOLVED

The goal of the CUE project. now discontinued, has been to provide an extensible environment for building and using integrated interactive computer services. In CUE, there are no boundaries between, say. the electronic mail service and the automatic calendar service. Consider the following task:

sending a message to everyone in my 3:30 meeting telling them I can't make it

Accomplishing this task requires a common understanding of the services involved: essentially, one must know how to find a particular meeting in the calendar system framework, access its attendees, and convert them to valid addressees in the message system framework. Unless the basis for such understanding has been designed into the system from the very beginning, automatic interservice interaction at this level is impossible. In current systems, almost all of which consist of totally separate service programs or "subsystems," the necessary *interservice understanding* is completely outside of the design framework. If a user wants to perform a task like the one above, he must interact with each service program separately, doing his own data conversions (usually via a text editor). In short, given current software environments, the problems of such a priori program planning and implementation to allow for interservice interaction are intractable. Tasks like the one above cannot be performed without a great deal of painful user involvement.

Automatic interservice interaction requires an environment in which individual services can make assumptions about the properties and the behavior of other services in the system. These assumptions must relate to data structures, functional capability, and state information. The objective of the CUE project has been to produce such a consistent underlying environment (CUE), in which users can interact with the system without regard to service boundaries.

13.2 GOALS AND APPROACH

The key element of the CUE strategy is to have a semantic model of how each program element (function or data structure) fits in with the other program elements in the system. That is, when a "send message" function is defined, CUE must have a model of how the elements of a message (e.g., the "user" in the addressee field) are already being used in the system. It must also know that "sending" involves transmitting information from one place to another, that the information must be made accessible at the destination, and so on. CUE can then use this knowledge to ensure that each new program element fits in appropriately with, and can be used as a part of, overall system functionality. CUE must also act as an execution environment which supports the sharing (or coercion) of data structures and the uniformity of function invocation necessary for interservice interaction.

To take full advantage of CUE's underlying consistency. the user interface must facilitate two activities: (1) sharing data from various services. and (2) combining functions from various services. All interaction with CUE is through a multi-windowed full-screen editor. Such an interface is ideal for encouraging data sharing, because it allows the user (through movement of a graphic cursor) to select any displayed information and to insert a copy of it at any position on the display. Functional combination is encouraged by allowing arbitrary functions to be applied to any selected region of a window and by supporting a functional style of computing in the command language.

13.2.1 Research Issues

The CUE project has addressed the following research issues:

- specifying a common data definition substrate;
- representing the relationships among structures and functions of different services;
- defining the user interface;
- achieving the correct level of modularity (grain size) for service functions;
- finding the correct level of interaction with a knowledge-based user interface (the Consul system); and
- providing transportability to maximize the impact of CUE on personal workstations.

13.2.1.1 Data definition

The problem of common data definition is well known in computer science; several kinds of solutions have been or are being investigated (e.g., [4], [13]). Work on abstract data types [2] has also had a very strong influence on CUE. This technique was appealing because of its compatibility with the requirements of knowledge-based modeling and because it provides a well-defined and consistent view of data structures.

13.2.1.2 Modeling service relationships

A few current systems (e.g., UNIX) contain primitive mechanisms for combining the functional elements and data structures of different interactive services. However, there is never a semantic model of the relationships between different service entities. Thus, the systems have no representation of the basis for commonality between, say, a message service and a calendar service. Existing systems can therefore never go very far toward ensuring integration: they must rely on each service builder to think of the necessary interrelationships among his service and others. and to design in enough flexibility to handle any interactions that might occur.

CUE avoids this total reliance on the service builder by providing a semantic model that links the functions and data structures of one service to those of others. CUE provides an acquisition mechanism that associates the program elements of new services with the constructs of this model. Research on knowledge representation and acquisition in the Consul project provides already well-developed techniques for modeling the characteristics of interactive systems [8, 12]. The CUE project uses and extends Consul's model to provide the independent semantic description that allows the system to relate the data and functions of different services.

13.2.1.3 The user interface

CUE's user interface focuses on techniques which facilitate interservice sharing, promote a consistent user view of the system, and guide the user through interactions. A number of existing techniques and extensions to them have influenced the design of the user interface.

Multi-windowed full-screen editors are a proven technique for coordinating and sharing information between files. The benefits of supplying an omnipresent editor were demonstrated in the SIGMA message system [11]. The extension of this approach to a multi-service environment encourages a consistent user view by providing a uniform mechanism for the examination and modification of data. A functional style of computing [1] is used to further encourage the combination of functions from different services.

Extensive use of forms and menus helps the user specify his task. The value of such an approach has been demonstrated in commercial systems like the Xerox Star as well as in research efforts such as COUSIN [3] and MENUNIX [9]. The main difference between the CUE approach and that of COUSIN or MENUNIX is the integration of services and the resulting pervasiveness of the user interface. All data, not just command lines, are handled through the same user interface in CUE. Furthermore, while the physical appearances of the CUE interface and the COUSIN interface may resemble each other, they are generated by very different processes. In CUE, after acquisition of a function, the user interface is automatically updated to make that function available and to make the invocation of it consistent with similar functions.

CUE offers multiple levels of help. Menus and forms are a first level, because they make it clear to the user what functionality is available. User inquiries about valid parameterizations of functions are also handled through menus. For example, if the user wants to delete a file, the request for valid parameterizations lists all of the files he can delete. Explanation text is an integral part of the user interface. Since CUE and Consul use the same knowledge base, CUE can use Consul's explanation facility to generate this text for service constructs automatically. In addition, an integral part of the CUE project is provision for communication between the CUE and Consul systems, thus giving the CUE user access to Consul's natural language help and explanation facility for very specific help requests.

CUE research on the user interface has also included experiments that investigate the most user-efficient ways of using bit-mapped graphics, electronic pointing devices, speech, and other state-of-the-art interaction media. The CUE user interface is designed to be very tailorable and device independent. The user is given great flexibility in modifying the interaction protocols, allowing experiments with different methods to be conducted and providing the user with modifiability to meet individual needs and preferences. Device independence is supported with multiple input channels available to the user at all times. The user decides which input device is most convenient and simply uses it. Mixed-media input is supported, so that parts of a command may be spoken while other parts may be typed and still other parts pointed to on the screen. Multimedia interfaces are a new development, and it takes some amount of experience to learn how to best use them.

13.2.1.4 Grain size

Ensuring the appropriate functional grain size for each service requires checking with respect to the semantic model and with respect to the other functions in the system. CUE uses the model to ensure that the size of each new function makes *semantic* sense (i.e., that the function works on the right kind of objects—for example, that a function purporting to be a "send" puts message data structures into mailbox data structures). In addition, CUE uses its knowledge of the other functions in the system to ensure that each new function makes *structural* sense (e.g., that there are other functions for accessing the mailbox data structure and for composing the messages).

13.2.1.5 Consul interaction

CUE's user interface is designed to interact with Consul to provide natural language request handling and to support the generation of specific help for the user's problems. Practical application of Consul is at least several years off; however, CUE is tied into Consul research, and as the technology becomes practical CUE offers it to the user. When Consul comes on line, the CUE user can type sentences such as, "Allow anyone to read this file," or "How can I recover from this error?" CUE notices that these are not command requests, realizes that they might be natural language requests, and passes them to Consul. Consul is able to process sentences such as these only if it has a model of the current environment. CUE must supply Consul with enough information to build this model. The CUE and Consul projects have worked together to determine how much of the state should be modeled in Consul and what mechanism Consul uses to reference or access information that is not explicitly modeled. The projects have also determined an appropriate formalism for efficient communication at two levels: knowledge base operations and procedure invocation (inference procedures in Consul and service procedures in CUE). From the operating system viewpoint, these could both be interprocess communications that would be covered by the appropriate protocols (see [10]).

13.3 SCIENTIFIC PROGRESS

The CUE project has continued to rely on the formalism of abstract data types. This formalism provides the basis for the common data definition substrate as well as the modeling of data structures. Reliance on abstract data types also provides some of the answers with respect to the user interface, because they provide a model of data that is an abstraction of the way users think about it. Combined with generic functions, abstract data types simplify both the modeling in Consul and the communication required between CUE and Consul. Finally, abstract data types and interprocess communication combine to give CUE a methodology for both integrating existing services into the CUE environment and distributing functionality across a network.

The strategy of using abstract data types as the basis for modeling, data definition, and the user interface has been tested by generating a demonstration system. The demonstration shows how a new data structure is accepted by the system, how that data structure is modeled in the system, and how it is linked to the domain semantics model. All of this happens via an acquisition dialogue. The demonstration relies on a model of programming semantics that was developed over the past year and on a preliminary version of the CUE interaction environment. The acquisition strategy and processing were jointly developed with the Consul project.

The CUE project has continued to develop a generalized forms package and a strategy for employing it in the system. All information in CUE is presented via a generalized form. The following generalizations are made in the CUE forms package:

- Forms are hierarchically organized—the fields of a form are generally other forms, not simply text strings.
- Classes of forms can be defined to encourage consistency across forms and to simplify their definition.
- Form generation and placement protocols are user modifiable.

- Forms define the interaction between the user and the form as well as the graphic attributes of the form.
- Forms may have functions attached to them which perform actions such as testing data. controlling the flow of the interaction, and computing values.
- Forms may use multimedia communications.

The forms package can now be used to display structured data. A preliminary version of an interaction environment and editor has also been implemented.

The publications and presentations of the CUE project include the following:

- presentation of a paper on the CUE/Consul interface at the CHI '83 conference [5];
- presentation of a paper on the use of multiple command modes at the 1984 Conference on Intelligent Systems and Machines [6];
- publication of a CUE working paper describing the forms package [7].

13.4 IMPACT

Many current applications are underutilized because they are not well integrated with other systems with which they naturally interact. Recent successes in integrating just three services (spread sheets, graphics, and databases[1]) have demonstrated the impact that integration can have. However, integration beyond a few services is impractical without a methodology to oversee the installation of new services into the environment. The knowledge-based approach of CUE can provide the basis for enforcing integration.

CUE has provided some insights into solving two of the major problems faced in the development of applications:

- designers would like to minimize inconsistencies in the interface as the user moves from one application to another, and
- roughly 60 percent of current development efforts are committed to the generation of the user interface.

With the advent of more sophisticated interactive devices, the time spent on the user interface and the inconsistencies found across applications will only increase. A methodology is needed to automate much of the responsibility for the generation of the user interface. This problem can also be solved by CUE's knowledge-based approach.

Personal workstations are the way of the future. and CUE has influenced the development of this important technology. Personal workstations will support significant local processing to avoid delays seen in current time-sharing systems. and will use bit-mapped graphics to improve the presentation of information to the user. The hardware for these workstations is now coming on the market from several sources. However. the software available on current workstations. while advanced in some respects. is inadequate for the long term because of its user interface and lack of support for service integration.

Software which intelligently uses the advanced graphics capabilities of current workstations,

[1]The commercially available systems. Context MBA and Lotus 1-2-3.

provides a consistent and understandable interface, and allows interservice interaction (i.e., data sharing and functional cooperation) must be developed to prevent the entrenchment of the current software in this new and important computing environment. The consequence of not attending to these problems is a system which will only exacerbate the problems now faced. Although CUE has been discontinued, the ideas explored by CUE are being used to form the basis for the new Formalized System Development project. This new project will be the delivery mechanism for CUE.

REFERENCES

1. Backus, J., "Function-level computing." *IEEE Spectrum* 19, (8), August 1982, 22–28.

2. Dahl, O. J., E. W. Dijkstra, and C. A. R. Hoare. *A.P.I.C. Studies in Data Processing.* Volume 8: *Structured Programming.* Academic Press, 1972.

3. Hayes, P. J., "Cooperative command interaction through the COUSIN system," in *International Conference on Man/Machine Systems*, Manchester, July 1982.

4. NASA, *IPAD: Integrated Programs For Aerospace-Vehicle Design*, NASA, 1980.

5. Kaczmarek, T., W. Mark, and N. Sondheimer, "The Consul/CUE interface: An integrated interactive environment," in *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 98–102, ACM/SIGCHI, December 1983.

6. Kaczmarek, T., and N. Sondheimer, "Man-machine interface modes: Variety and integration," in *Proceeedings of the 1984 Conference on Intelligent Systems and Machines*, Rochester, Michigan, April 1984.

7. Kaczmarek, T., "CUE Forms Description," CUE Working Paper, 1984.

8. Mark, W., "Representation and inference in the Consul system," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, IJCAI, 1981.

9. Perlman, G., *Two Papers in Cognitive Engineering: The Design of an Interface to a Programming System, and MENUNIX: A Menu-Based Interface to UNIX (User Manual)*, National Technical Information Service, AD/A108 929, November 1981.

10. Rashid, R., *An Inter-Process Communication Facility for UNIX*, Canegie-Mellon University, Technical Report, March 1980.

11. Stotz, R., et al., *SIGMA Final Report*, USC/Information Sciences Institute, RR–82–94 and RR–82-95, 1982.

12. Wilczynski, D., "Knowledge acquisition in the Consul system," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, IJCAI, 1981.

13. Yntema, D. B., *The Cambridge Project: Computer Methods for Analysis and Modeling of Complex Systems*, Rome Air Development Center, Technical Report, 1973.

14. KNOWLEDGE DELIVERY RESEARCH

Research Staff:
William C. Mann
Michael Fehling
Barbara Fox
Michael Halliday
Christian Matthiessen
Sandra Thompson

Support Staff:
Lisa Trentham

14.1 PROBLEM BEING SOLVED

The general practical goal of the Knowledge Delivery project is to allow computers to express information in appropriate, easily read, multiparagraph English text.[1] This goal has never been accomplished with any generality. Computers have generated text only for very narrow domains of subject matter, and always with many handcrafted computational parts.[2]

In order to reach this practical goal, several difficult theoretical problems must be solved:

1. The *vocabulary* must be characterized, and its relation to available knowledge in the computer must be specified in detail.
2. The *grammar of English* must be characterized, especially the close alternatives among grammatical constructions and the relationships between particular grammatical constructions and particular effects to be produced by the generated language.
3. The reasons for selecting particular *grammatical constructions* must be characterized.
4. The reasons for expressing information in a particular *order* must be identified.
5. The reasons for *grouping ideas together* in text, and the specific effects of doing so, must be identified.
6. The reasons for *choosing to express or withhold* particular information must be identified. (This has been a dominant problem in past work on text generation; computers carry so much information on particular topics that very strong selectivity is needed.)
7. People (and eventually machines) produce text for definite reasons. An appropriate *notation for text goals* is needed.
8. The processes that produce text are extremely complex. They require appropriately *complex and flexible control methods.*

The fields that contribute the most to this investigation are computer science and linguistics. The linguistic knowledge of the ways language goes together and produces effects is particularly vital.

Nearly all linguistic investigations are done in a way that characterizes actual text in terms of particular abstractions. They describe text, but do not attempt to say how text might be constructed. Because description is a kind of abstraction, it loses information. Many of the regularities and patterns of good text are not accounted for by linguistic description, yet they are required when good text is constructed. They must therefore be accounted for in any comprehensive text generation process. These undescribed gaps are a significant source of difficulty in text generation research.

[1]The need exists just as much for languages other than English, but English is in many ways the opportunistic target.

[2]There are long computer-generated texts created by a fill-in-the-blanks method, but there the real author is a person.

Their presence helps explain why progress of the field has been so slow, and why so much precomputational theoretical work must be done in creating an effective text generator.

The complexity of the problem makes it important to consider a second kind of goal: Research results must somehow be carried forward from one investigation to another. Journal articles and other publications are necessary, but inadequate, since so much of the created knowledge resides in the details of particular computer programs. The programs themselves must be passed around the research community in a usable form. For the community as a whole, this is a largely unsolved problem. For this project, we have made some progress by creating an elaborate portable research grammar of English.

14.2 GOALS AND APPROACH

To reach the general goals of the project, we have focused on more specific project goals:

1. Designing an overall program structure in which grammar, programmed knowledge of discourse, and other components can be appropriately related and controlled. The approach, starting with the existing Penman design, is to observe the information dependencies found in increasingly complex worked examples, whether programmed or hand-simulated, and to continually update the design to provide for these dependencies.
2. Creating a computational grammar of English for text generation that is transferable from one research group and purpose to another. The approach, starting with the existing Nigel implementation, is to expand the implementation to cover new cases and remove bugs, preparatory to circulating the system to other researchers.
3. Creating an explicit theory of discourse structure, suitable as a basis for computer program implementation. The approach is to create a descriptive theory that covers actual texts, and then expand it to be a constructive theory that can be followed to construct simple texts in hand simulation, preparatory to programming.

14.3 SCIENTIFIC PROGRESS

The accomplishments of the project can be divided into three areas, which correspond to the goals and approaches above and are described in the three subsections immediately below.

14.3.1 Penman: An Evolving Design

The project's text generation system is called Penman. Parts of Penman are programmed, and other parts are at various levels of specification as program designs. One of Penman's modules is Nigel, the grammar. This discussion of Penman concerns its overall control structure; the next section describes Nigel.

The most common sort of generator design moves the information through a pipeline. It starts with an initiating request or goal, then proceeds to a search for things to say, then to an organizing phase, and finally to production as a set of sentences. This design has several deficiencies, which are corrected in the Penman design.

1. One deficiency concerns *the relationship between search and presentation*. If the search for things to say is completed before the organization phase (as in the pipeline design), search cannot be used to solve problems of text organization such as the need for background information, the need for evidence for doubtful points, or the need to concede away some apparent refutation. Early versions of Penman did all content discovery before starting to organize the material, incorporating this defect.

In the current Penman design, discovery of things to say is subordinated to text planning. As part of the organization process there are specific limited searches for particular kinds of information. This design is more efficient and more effective.

2. A second control design problem concerns *the degree to which the effects of particular decisions can be anticipated.* As in every complex system, some consequences of decisions are effectively unpredictable. In the pipeline design, decisions in early modules are never assessed or corrected. The text produced by pipeline text generation systems thus contains characteristic defects.

   In Penman, a specific module is devoted to text improvement. It assesses the text, checking for recognized patterns of defect. It then selects a high-priority defect to correct, modifies the text plan accordingly, and regenerates the changed portions. The text is complete only when this "hindsight module" cannot produce improvements. This approach to control overcomes the problem of weak anticipation of the consequences of decisions.

14.3.2 The Nigel Text Generation Grammar

When this project was begun, there was no single preeminent variety of grammar for text generation. There are many grammatical formalisms, but only a few relate grammatical structure to the effects the language produces. The Systemic Functional Grammar framework of Systemic Linguistics was chosen for this work. Fortunately, shortly after the Systemic framework was chosen, its founder (Michael Halliday) joined the project as a consultant.

At the beginning of this research, Systemic Linguistics had described many portions of English in separate fragments of grammar, but (aside from Hudson's work [1], which we did not follow in detail) there was no comprehensive, consistent treatment in a single notation. The grammar we have built, called Nigel, is now the most comprehensive grammar of English for text generation, necessarily in a single notation since it is programmed.

Before Nigel was built, systemicists had not yet agreed on the details of how grammatical structures are constructed; Nigel has a more explicit and formal notation for structure building than any predecessor.

Beyond these refinements of the grammatical notation, the work on Nigel has extended the Systemic framework so that its semantics is much more explicit than before. Systemic grammar is organized around choice among alternatives. In the development of a Systemic grammar, sets of closely related alternatives are identified and the patterns of alternatives are described in Systemic notation. However, describing the alternatives does not include describing the circumstances under which each particular alternative is chosen. Although those circumstances are often described informally, the grammar's relation to situations of language use is not made very explicit in that way.

To make the choice processes explicit and programmable, as part of the development of Nigel, a new formal notation for choice among alternatives has been created. This notation and the choice procedures expressed in it are called *Inquiry Semantics.* In addition to facilitating programming, Inquiry Semantics is an extension of the Systemic Linguistics framework.

In a major test of the Inquiry Semantics framework, choice procedures have been created for all of the systems of Nigel. The result of this extensive effort is a major explication of the uses of the grammatical alternatives of English.

With this approach to grammar, the generated language can be very well adapted to running text. Other approaches concentrate far more on isolated sentences, but Systemic Linguistics has built grammars which have extensive provisions for making text flow smoothly. This will be particularly helpful in generating multiparagraph text.

Among computational grammars of natural languages such as English, Nigel is one of the most comprehensive. We therefore expect that, as we disseminate it to appropriate research centers and developers, a wide range of technical and practical goals will be facilitated.

14.3.3 Rhetorical Structure Theory

It is generally recognized that computer-generated text must be *planned*. The knowledge of text planning is currently rather primitive, partly because related academic topic areas (such as discourse analysis) have focused exclusively on analysis and description, not on synthesis.

For a computer to be able to plan text, there must be an underlying theory of text which is the basis for the program. At the beginning of this research, no suitable theories existed. What was known was too vague and analytically oriented to use as the basis for program design.

We have created a new body of theory, called *Rhetorical Structure Theory* (RST), which represents the structure of written monologues such as reports, magazine articles, instructions, and other expository kinds of text. The theory is currently descriptive rather than constructive, i.e., it does not yet show how to do all the steps of text planning. It does specify what a text plan must contain, and informal experiments have shown that it is useful as a basis for text creation by people. On a large scale, it has been used to plan technical reports and conference papers. On a smaller scale, it has been used to show how particular existing texts could have been created by a programmed generator. These informal experiments demonstrate a level of definiteness and planfulness beyond what the published art had previously achieved, and they show that RST is suitable as a text planning design base.

14.3.4 Recent Project Publications

Recent publications of the Knowledge Delivery project are described below. Complete references for these publications are given in the bibliography.

1. *An Overview of the Penman Text Generation System* [2]. Describes Penman's general design and identifies operations that are required to support general text generation.
2. *Systemic Encounters with Computation* [3]. A historical survey of all of the computational work done in the Systemic framework.
3. *Inquiry Semantics: A Functional Semantics of Natural Language Grammar* [4]. Describes the interactions between Nigel and its informational environment, and shows how these interactions determine what Nigel says.
4. *A Linguistic Overview of the Nigel Text Generation Grammar* [5]. Describes Nigel for a linguistic audience.
5. *An Introduction to the Nigel Text Generation Grammar* [6]. Basic orientation to Nigel's capabilities, notation, and content.
6. *Relational Propositions in Discourse* [9]. Describes varieties of assertions that arise in running discourse but not in isolated sentences. Establishes several categories of assertions that can be made from discourse structure alone, without explicit signals.
7. *Choosing Tense in English* [10]. A Systemic treatment of tense in English, based on Nigel.

8. *Choosing Primary Tense in English* [11]. A journal article containing a Systemic treatment of English primary tense. based on [10].
9. *Choosing Tense in English* [12]. A full description of Nigel's treatment of English tense, expanding on [10] to include justifications, additional detail, and comparison with other approaches.
10. *The Systemic Framework in Text Generation: Nigel* [13]. A linguistically oriented description of how the general desiderata of the Systemic framework interact with the text generation task. taking Nigel as the central example.
11. *Discourse Structures for Text Generation* [7]. Describes Rhetorical Structure Theory as a basis for text planning. and compares it with prior computational text planning work.
12. *What's in Nigel: 1* [14]. A brief introduction to the content of Nigel for functional linguists.
13. *The Realization Operators of the Nigel Grammar* [8]. A brief introduction to Nigel's methods of structure building for functional linguists.

14.4 IMPACT

This project has produced several of the major technological prerequisites to creating a general-purpose. programmed generator of English text. The impact of this work has been in computational linguistics, artificial intelligence, and linguistics. In computational linguistics, other research groups will use the Nigel grammar as a component in future research work on text generation systems. They have seen its definitional framework and its treatment of English as particularly attractive as a base for research in human–computer interfaces. In artificial intelligence, some of the representational gaps which Nigel has revealed are being studied with a view to filling them. In linguistics, Nigel's extension of the Systemic framework is contributing to new studies of language function.

When Nigel is made available as a research resource. researchers in text generation will be able to concentrate more attention on areas other than sentential fluency. Nigel is large enough that, when some significant gap in its grammar is found, it will be preferable to add the new construction to Nigel rather than starting over on a domain-specific grammar. Development of the grammar will become cumulative. with a resulting increase in research productivity simply because discoveries are retained and propagated.

Part of the eventual nonresearch impact of the work will be to make computer interfaces easier to construct and use. Another part of the impact will be to make new kinds of computer applications practical—those which require easy expression of information to computer-naive people. This creation of new application possibilities may be even more significant than facilitation of existing ones.

14.5 FUTURE WORK

In follow-on projects already under way. we will distribute the Nigel grammar to other research organizations. and will build several text generators for specific purposes. We will also work in the areas of lexical selection. noun phrase planning. sentence planning. interfacing Nigel to various programmed knowledge notations. text planning. text brevity techniques. and appropriate representation of the static and dynamic knowledge of the reader. Each of these kinds of technical development will enhance the quality. flexibility, or expressive power of the system.

REFERENCES

1. Hudson, R. A., *North-Holland Linguistic Series.* Volume 4: *English Complex Sentences,* North-Holland, London and Amsterdam, 1971.

2. Mann, W. C., "An overview of the Penman text generation system," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 261-265. AAAI, August 1983. Also appears as USC/Information Sciences Institute, RR-83-114.

3. Mann, W. C., "Systemic encounters with computation," *Network: News, Views and Reviews in Systemic Linguistics and Related Areas*, (5), May 1983, 27-33.

4. Mann, W. C., "Inquiry semantics: A functional semantics of natural language grammar," in *Proceedings of the First Annual Conference*, Association for Computational Linguistics, European Chapter, September 1983.

5. Mann, W. C., "A linguistic overview of the Nigel text generation grammar," in *Proceedings of the Xth International LACUS Forum*, Linguistic Association of Canada and the United States, Quebec City, Quebec, Canada, August 1983.

6. Mann, W. C., "An introduction to the Nigel text generation grammar," in *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, RR-83-105, February 1983. This paper will also appear in a forthcoming volume of the *Advances in Discourse Processes Series*, R. Freedle (ed.): *Systemic Perspectives on Discourse: Selected Theoretical Papers from the 9th International Systemic Workshop,* to be published by Ablex.

7. Mann, W. C., *Discourse Structures for Text Generation*, USC/Information Sciences Institute. RR-84-127, February 1984. Also appeared in the proceedings of the 1984 Coling/ACL conference, July 1984.

8. Mann, W. C., and C. M. I. M. Matthiessen, "The realization operators of the Nigel grammar," *Network: News, Views and Reviews in Systemic Linguistics and Related Areas*, (7), 1984, to appear.

9. Mann, W. C., and S. A. Thompson, *Relational Propositions in Discourse*, USC/Information Sciences Institute, RR-83-115, July 1983. To appear in *Discourse Processes.*

10. Matthiessen, C. M. I. M., Choosing Tense in English, Master's thesis, University of California at Los Angeles, 1983.

11. Matthiessen, C. M. I. M., "Choosing primary tense in English." *Studies in Language* 7, (3), 1983.

12. Matthiessen, C. M. I. M., Choosing Tense in English, 1983.

13. Matthiessen, C. M. I. M., "The systemic framework in text generation: Nigel," in *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, RR-83-105, 1983. This paper will also appear in a forthcoming volume of the *Advances in Discourse Processes Series*, R. Freedle (ed.): *Systemic Perspectives on Discourse: Selected Theoretical Papers from the 9th International Systemic Workshop,* to be published by Ablex.

14. Matthiessen, C. M. I. M., "What's in Nigel: 1," *Network: News, Views and Reviews in Systemic Linguistics and Related Areas*, (6), April 1984, 36-44.

INFORMATION PROCESSING DIVISION

The Information Processing Division (IPD) conducts support projects that provide established computing tools to researchers both at ISI and across the Defense Data Network (DDN). The IPD also includes projects that actively pursue, develop, and integrate new computing tools and environments for eventual inclusion into the set of supported tools of the research community. The IPD is responsible for several architectures and operating systems, as well as their networks. The following machine types are supported:

- Digital Equipment Corporation (DEC) - VAX 11/780, VAX 11/750, KL2060T, and KL1090T
- Xerox - 1100
- Symbolics - 3600
- Sun Microsystems - 68010
- IBM - PC, XT, and AT
- Hewlett-Packard workstations
- Apollo workstations

The Computer Research Support project provides reliable and sufficient computing facilities to the ISI (and a portion of the DDN) research and development communities on a 24-hour, 7-day schedule. The Information Processing Center (IPC) is primarily composed of six DEC 2060T, ten DEC VAX 11/750, and three DEC VAX 11/780 computers, as well as associated communications equipment and peripherals. The IPC also provides extensive support to the Design Center at Gunter Air Force Base in Alabama.

The New Computing Environment (NCE) project envisions local processing nodes (workstations and multiuser mainframes) connected via a local network to a set of central servers. The eventual goal for this project is to develop and make attractive (for researchers and other ISI computer users) the option of migrating from the supported multiuser mainframe systems to more modern personal computers, workstations, and stand-alone mainframes. NCE is also a testbed and an example to the computing research community of one possible solution to the problem of computer architecture migration.

The Exportable Server System (ESS) and Exportable Workstation Systems (EWS) projects are closely related. Together, they seek to develop and provide a sophisticated information system for the 1990s. As workstations, personal computers, and word processors become integrated into local area networks, the need for localized server centers arises. Gateways, large disk servers, long-haul network connections, and other expensive peripherals will be consolidated in these centers to provide proper environmental conditions and to allow the sharing of this hardware.

Using recent advances in artificial intelligence, computer science, and microelectronics, DARPA plans to create a new generation of "machine intelligence technology." Our Strategic Computing project will

- provide development computers for the Strategic Computing program,
- provide system integration as required,
- provide distribution mechanisms for disseminating the systems to the program participants, and
- define an architecture for system communications and resource sharing among the computers acquired.

DARPA, in cooperation with the World Wide Military Command Control System (WWMCCS) System Engineer (WSE) and the Strategic Air Command (SAC), has chartered a Strategic Command, Control, and Communications (C3) System Experiment. The C3 System Experiment Support project will provide an initial core of necessary facilities (ARPANET/MILNET access, host systems, various software tools, Network Services support, etc.) to allow SAC personnel to gain experience with current technology and to ensure the success of the broader experiment. We will install and maintain modems and 55 or more interactive CRT terminals, and will provide user training, system software training and support, and on-site maintenance of equipment.

The Interlisp project provides large address-space portable versions of Interlisp for the VAX and other hardware architectures, and is a self-sustaining support center for the Interlisp language. The current implementation of Interlisp, known as ISI-Interlisp, runs on the VAX family of computers, under both the UNIX and VMS operating systems. In addition to ongoing maintenance tasks, development efforts are planned for the future, including the porting of Interlisp to additional machine architectures.

The Department of Defense has adopted the Internet concept—the IP and TCP protocols in particular—as DoD-wide standards for their packet networks. The principal goal of the TCP/IP Implementation Support project at ISI is to install and maintain the software required to allow the interoperability of the ARPA-Internet's ever-growing collection of long-haul and local networks. This is an ambitious task and has already been extremely successful. ISI's Internet project and several groups at Bolt Beranek and Newman, Inc., continue to provide standards and testbeds for the protocols and gateways of the Internet.

15. COMPUTER RESEARCH SUPPORT

Director: Gary L. McGreal
Technical Assistant: Dale Russell
Liaison Gunter Facility: Dave Hale

Technical Staff:

Systems Software:
- Joel Goldberger
- David Bilkis
- Mike Butler
- William Moore
- Craig Rogers
- Craig Ward
- John Weber
- Tom Wisniewski

Hardware:
- Daniel Pederson
- Glen Gauthier
- Ramon Gonzalez
- Norman Jalbert
- Raymond Mason
- Jeff Rolinc
- Daniel Trentham
- Leo Yamanaka

Operations and Network Services:
- Stephen Dougherty
- Walt Edmison
- Steve Carroll
- Brent Gesch
- Roylene Jones
- Joseph Kemp
- Linda Sato
- Toby Stanley
- Mike Zonfrillo
- Vicki Gordon
- Sam Delatorre
- Anna-Lena Neches
- Wayne Tanner
- Christine Tomovich

Support Staff:
- Manon Levenberg
- Pat Thompson
- Vivian York

15.1 PROBLEMS BEING SOLVED

The Computer Research Support project is responsible for providing reliable computing facilities on a 24-hour, 7-day schedule to the ARPANET/MILNET research and development community. At the same time, the project makes available to ARPANET/MILNET users the latest upgrades and revisions of hardware and software. The project provides continuous computer center supervision and operation, and a full-time customer-service staff that is responsive to user inquiries. This project supports two computer installations the larger, main facility at ISI in Marina del Rey, and the Design Center at Gunter Air Force Base in Alabama.

15.2 GOALS AND APPROACHES

The ISI Information Processing Center provides support in four tightly interrelated, though distinct, areas: Hardware, System Software, Operations, and Network Services. The overall charter of the organization is to assure that the needs of the user community be addressed and protected as effectively as possible. All groups are concerned with the efficient use of the machines and with the security of the physical plant and all online information. The more specific responsibilities of each group are summarized below.

15.2.1 Hardware

To achieve a reliability goal of 98.7 percent scheduled uptime, preventive and remedial maintenance responsibilities are assigned to an in-house computer maintenance group. This group provides 20-hour, 5-day on-site coverage and on-call standby coverage after hours. To maintain the reliability goals, preventive maintenance is very closely controlled, and online diagnostics and analysis are emphasized. A primary component in the reliability and availability of the hardware is the physical environment in which it exists. Accordingly, a significant amount of time and resources is expended in ensuring that the best, most cost-effective environmental controls are used at all facilities that ISI services.

15.2.2 System Software

The software group's primary responsibility is the installation and maintenance of ISI's VMS, UNIX, and TOPS-20 operating systems and applications software. In order to provide maximum reliability, the group provides 24-hour, 7-day coverage to analyze system crashes and to provide appropriate repairs. The System Software group also installs, debugs, and modifies both the latest monitor versions and the associated subsystems available from the vendor.

15.2.3 Operations

The operations staff is responsible for operating the computers and overseeing the systems and the environment. There is 24-hour, 7-day on-site coverage at the Marina del Rey facility. One of the primary responsibilities of the group is to provide protection and backup for user files to insure the integrity of the data. This is achieved through a variety of means, including regularly scheduled full and incremental backups of all systems; permanent archivals of requested or infrequently accessed system and user files; tape storage and pointers to all information extant at the time of removal of directories from the various systems; and, perhaps most important, redundant offsite storage of all significant information active on the disk structures or existing on tape within the facility. When a problem occurs, the on-duty staff isolates it and takes appropriate action. On the night and weekend shifts, the operations staff responds directly to user inquiries. Proper training, experience, and familiarity with the environment are especially important.

15.2.4 Network Services

Network Services, the ISI customer service group, provides a two-way communication link between the users and the entire support staff. This is accomplished by maintaining a 12-hour, 5-day on-duty staff for prompt problem resolution and rapid information exchange, both online and by telephone. The group offers introductory training in the use of both the hardware and software tools available on the ISI systems, as well as providing documentation for new users of the ARPANET/MILNET. Network Services also assists in the formulation of user training programs for large, ARPANET-based military experiments, such as the U.S. Army XVIII Airborne Division Army Data Distribution System at Fort Bragg, North Carolina, and the C3/ARPANET Experiment at Headquarters, Strategic Air Command, at Offutt Air Force Base, Nebraska. Appropriate documentation is constantly being generated and distributed to individual users, as well as to remote user-group liaison personnel; this documentation ranges from simple, novice-level explanations to more technical information suitable for experienced users. In accordance with IPTO guidelines, the customer-service group provides regular accounting data to DARPA.

15.3 PROGRESS

The past year has been a very successful one for the Computer Center. Average uptimes were 99+ percent, and several major transitions have occurred without great difficulty and, more important, with virtual transparency to the user community. We continued to appreciate and reap the benefits of the physical modernization effort completed in late 1982 and early 1983 which resulted in essentially no environmental system outages.

15.3.1 Hardware Additions

Over the past three years the ISI Computer Center has undergone a major change in direction. Previously dominated by DEC PDP-10 computers, the facility was a relatively simple support environment consisting of a uniform collection of hardware and software. The current collection of equipment reflects the New Computing Environment project's plan to move towards personal workstations and rear-end servers. The new collection of systems creates a substantially more complex support problem, involving two local networks (in addition to the ARPANET/MILNET), ten different processors, and seven operating systems:

Quantity:	*Manufacturer:*	*Model:*	*Operating System:*
6	DEC	PDP-10	TOPS-20
2	DEC	VAX 11/780	VMS
1	DEC	VAX 11/780	UNIX
7	DEC	VAX 11/750	VMS
3	DEC	VAX 11/750	UNIX
8	XEROX	8010	STAR/MESA
3	XEROX	1100	INTERLISP BASED
3	3-RIVERS	PERQ	PERQ
1	SYMBOLICS	LM-2	QLISP BASED
5	SYMBOLICS	3600	QLISP BASED
3	SUN	100/120/150	UNIX
30	IBM	PC	MS-DOS

In addition, another DEC PDP-10 is supported by the ISI staff at Gunter AFB. A map of the major pieces of the current local hardware configuration is shown in Figure 15-1.

15.3.2 System Software Enhancements

During this reporting period, a major effort centered upon making the above collection of hardware more compatible from a software point of view. The major work involved fully debugging TCP/IP for the various operating systems as well as writing and installing new software for hardware which had either no TCP/IP or no other robust method of communicating with the other systems. One of the most notable of these efforts focused on the IBM PC.

Another major effort involved the conversion or upgrade of major operating system releases themselves. The change with the largest impact on the user community was the upgrade to UNIX version 4.2 BSD (Berkeley Software Development) and to VMS version 3.6.

During this reporting period, with the transition of major hardware subsystems on the TOPS-20 machines—notably the phase out of the CALCOMP disk and tape systems to DEC RP07 disks and TU78 tapedrives—the software group had to make major modifications to the system monitor.

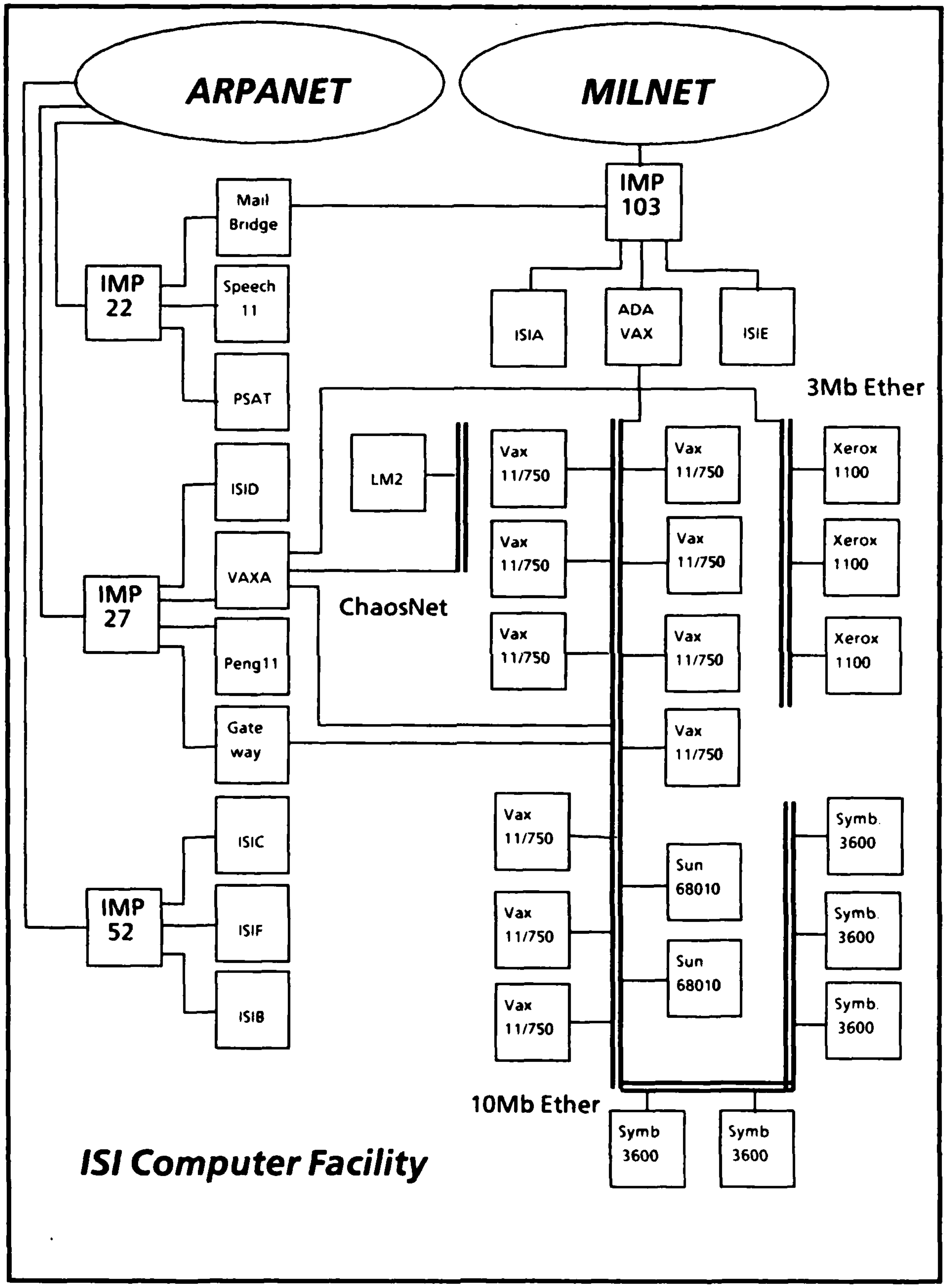


Figure 15-1: Diagram of local ISI ARPANET facility

Other significant work over the last year included accounting systems enhancements on TOPS-20, entirely new accounting systems on UNIX and VMS. and numerous bug fixes on many System and user subsystem programs.

15.4 IMPACT

ISI's computer centers provide ARPANET/MILNET cycles 24 hours a day, 7 days a week to the Strategic Air Command, Gunter Air Force Base, and Fort Bragg, as well as to the contractors, researchers, and administrators of militarily significant research coordinated out of the DARPA office in Washington, D.C. In addition to supplying machine time and directed Network Services support, this project continues to provide substantial additional support in the following areas:

- training. documentation, and modifications as requested by user groups for NLS;
- support for the production of AFM 67-1 with NLS;
- creation and support of specific staff scheduling programs for SAC;
- general accounting information on an as-needed basis for workload analysis on the various machines;
- rapid response to user requests; and
- tailored manual and automated security awareness and tracking.

15.5 FUTURE WORK

The Computer Research Support project will continue to provide computing service to the DARPA research community, provide and support software packages for the ARPANET/MILNET community, and offer a program of technology transfer and user education through the Network Services group.

Some specific activities are planned for the next reporting period:

- A gradual transition of local ISI staff to the NCE (New Computing Environment), resulting in the availibility of additional TOPS-20 resources for the wider military community.
- Installation of VMS version 4.0 on our VAX computers running VMS.
- Installation of the portions of TOPS-20 version 6.0 which will be of use to the ISI communities.

16. EXPORTABLE SERVER SYSTEM

***Research Staff*:**
Gary McGreal
Daniel Pederson

16.1 PROBLEM BEING SOLVED

A need for localized server centers arises as workstations, personal computers, and word processors become integrated into local area networks. Gateways, large disk servers, long-haul network connections, and other expensive peripherals will be consolidated in these centers to allow sharing of this hardware as well as proper environmental conditions.

The Exportable Server System project is associated with the Exportable Workstation Systems (EWS) project. Together, these efforts seek to develop and provide a sophisticated information system for the 1990s.

16.2 GOALS AND APPROACH

A principal goal of the Exportable Server System is to provide services of high reliability to the workstations on the local area network. It will probably not be economically feasible for onsite technicians to be available for most environments of this type: thus there is a need for highly reliable or redundant services which can be controlled from a remote service center.

The current Server System design consists of a modern computer room facility (albeit on a small scale) co-located at the remote workstation site. The air conditioning system for this remote service center will be fully redundant, with two dual-compressor machines installed in the computer room. Both will have twin hot-gas and liquid-return lines piped to air-cooled condensers installed on the roof. Condenser fan motors will have speed controls to provide control of discharge pressure under varying weather conditions. An alarm system and sensors will be installed to detect malfunctions and water leakage. Compressor and reheat sequencing will be automatic. Sensors and a feedback and control system will assure maintenance of proper room temperature and humidity levels. Twelve-inch raised flooring will provide a conduit for air movement throughout the room.

The facility will have an automatic fire detection and Halon dispersal system. A card key security system and closed-circuit TV system will control access to the facility. A solid-state power conditioning center will assure clean power to the facility. Power distribution units will provide electrical isolation and metering of system input power.

16.3 PROGRESS

Work is being carried out in five phases. During the first phase:

- walls will be installed with vapor barrier and S.T.C. 45 soundproofing;
- 220 amp 277/480 volt service will be run from existing building riser;
- 200 amp MLO panel 277/480 volt, 3 phase will be installed;
- 30 KVA 3 phase transformer with 100 amp MCB panel. 120/208 volt will be installed;

- branch circuits for floor power, heat pump, Halon system, air conditioning, and printers will be installed.

During the second phase:

- floor will be sealed with polyurethane sealer;
- non-flaking vinyl acoustical tile will be installed;
- card key cypher lock will be installed on main entrance and machine room;
- raised floor will be installed.

During the third phase:

- two Liebert FH199 air-cooled computer room air conditioning units with compressor sequence switch, mode alert, auto flush, auto restart, firestat, and liquitec sensors will be installed;
- two roof top condenser units will be installed;
- condensate drain lines will be installed;
- liebert SCA75C (75KVA) power conditioning unit will be installed;
- Halon 1301 fire detection and extinguishing system will be installed.

During the fourth and fifth phases, painting and finishes will be completed and computers and communications equipment will be installed. Work is in progress and on schedule. Completion of all five phases is scheduled for December 1, 1984.

16.4 IMPACT

In the military and general government office environment, there is a need to provide high-quality computer services at low cost. The Exportable Server System provides a model for allowing distributed services in a typical office environment. The design minimizes operational costs by providing a high degree of automatic environmental control. Further, it provides a testbed for the evaluation of concepts in remote maintenance technology.

16.5 FUTURE WORK

Once the completed center is in place, it will be operated as a server facility in support of the Exportable Workstation effort. It will also serve as the testbed for the Remote Maintenance Technology (RMT) project, which will seek to provide hardware support and system monitoring from a central operations center. Monitoring equipment to detect hardware, software, communications, and environmental problems will be located at the server site. System status reports will be carried to the central operations facility over the ARPANET. Based on monitoring and diagnostic data received over the network, central operations personnel will direct untrained remote staff to carry out simple maintenance procedures as required.

17. EXPORTABLE WORKSTATION SYSTEMS

Research Staff:
Joel Goldberger
Gary McGreal
Jim Koda
Dale Russell
Tom Wisniewski

Support Staff:
Manon Levenberg
Kay Freeman

17.1 PROBLEM BEING SOLVED

Fully capable personal workstations are no longer confined to computer science research centers and laboratories. They have advanced to the point that they are being introduced into office environments for use by non–computer professionals. Many of these professionals have only recently begun relying on the facilities available on large, centralized, time–shared mainframes accessed over the ARPA–Internet. They have also become accustomed to the reliability and ease of use of these large mainframes, which are supported by a staff of hardware and software specialists.

The transition from a time–shared environment to one of dedicated personal workstations requires careful planning and support. Every effort must be made to provide all of the facilities on which the current user community depends, and to guarantee that the new environment is as reliable and easy to use as the environment it replaces. The workstation environment is targeted for computer *consumers*, rather than computer *developers*; extra care is needed to free these users from having to perform their own support functions, such as software and hardware updates and debugging.

17.2 GOALS AND APPROACH

While the principal goal of this project is to install a workstation environment at a remote location (the DARPA–IPTO offices in Arlington, Virginia), it is also intended to provide a testbed for several other concepts and projects. The environment will be configured to accommodate a variety of workstation models with varying capabilities. This configuration will provide a minimum set of services for each workstation model:

- document preparation,
- mail handling,
- file storage and manipulation,
- administrative activities (spreadsheets, calendars, etc.).

These services will be provided either directly on the workstations, or on the central server (a DEC VAX–11/750). All of the workstations will be connected to a local Ethernet and to the ARPA–Internet via a BBN–maintained gateway and by equivalent code on the VAX that will function as a gateway in the event of failure.

The desire to free remote users from the responsibility of maintaining a computing environment has spawned the Remote Maintenance Technology project, which will investigate remote monitoring and maintenance techniques for this installation. In the next reporting period this project will develop hardware and software to remotely monitor and report on any anomalies (connectivity problems, hardware failures, environmental or power changes) and, in some cases, take corrective action.

17.3 PROGRESS

Ten SMI SUN workstations (a Motorola 68010-based UNIX system with graphics hardware and a large format display) have been installed at the DARPA-IPTO offices. The model presently installed has a local disk and is much more expensive than we had planned. We have made arrangements to trade this model in for a diskless workstation when it becomes available. We have also made arrangements for SMI to maintain these workstations until our remote monitoring and maintenance project is in place.

Various third-party software has been obtained and installed to provide basic services. These workstations are used by program managers and their assistants for the majority of the document preparation and financial modeling done in the office.

Due to delays in upgrading the existing physical plant, the VAX server has not been installed at the remote location. Presently, all file backup is being done from ISI via the ARPA-Internet, using the remote command facilities of Berkeley UNIX 4.2 BSD. Progress has been made on improving the reliability and usability of the mail system under the UNIX operating system, although most users still depend on the existing TOPS-20 systems for this facility.

Since this workstation was only recently introduced, a great many products for it are under development. We have obtained many of these and have others currently on order. Some early acquisitions are already obsolete; others will soon be replaced by new offerings.

17.4 IMPACT

The early acceptance of the workstations already in place and the ease of adaptation by previous time-sharing users clearly demonstrate the applicability of this new technology to office and administrative environments. This project has created considerable interest among our other user communities—military contractors in particular—as a means of providing increased computational capacity and enhanced functionality.

Everyone involved in this project agrees that dedicated personal workstations are superior to remote time-shared resources. The major advantages are greatly increased responsiveness of the system, elimination of delays caused by the Internet, and enhanced functionality available by virtue of the large format display and mouse pointing device offered on the workstation.

The ability to integrate additional workstation models into the environment and to have them function in an integrated manner will greatly enhance the portability and transfer potential of research efforts being carried out by most of the DARPA-sponsored groups around the country. The sponsors will be able to run the developed software on identical hardware without waiting for it to be adapted to another system.

17.5 FUTURE WORK

In the next year, we will improve the reliability and refine the functionality of the system. We will install more workstations, bringing the total population to twenty. The VAX server will be installed when construction of the facility is completed. We have replicated the remote environment at ISI so that we can proceed with the development of the software for both the VAX and the workstations well in advance of their installation at IPTO.

Our immediate goals are to install server software on the VAX to allow it to function as the file server for the workstation population. This will eventually eliminate the need for using workstations with local disks, allowing us to use the larger capacity peripherals of the VAX for online storage and tape backup.

We will continue to improve the mail system from the perspectives of both ine user interface and its internal functionality. The user interface will initially be made to resemble the more familiar TOPS-20 mail handlers (HERMES) and will then be adapted to support use of the windowing and mouse capabilities of the workstations. We will use the VAX as a central mail server, through which all mail entering and exiting the workstation environment will pass. This will simplify addressing requirements and will conceal the internal details of the environment from outside users. We also plan to use both the VAX and our existing TOPS-20 systems as a redundant mail cache for greater reliability.

We will continue to monitor the third-party software market and will obtain and install copies of all applicable software packages as they become available. This includes software being developed at other research centers as well. A specific task for the next year is to enhance a calendar system developed at the University of Wisconsin so that it can be used to coordinate multiple calendars in a distributed environment. We will also develop software to allow files to be converted among the various environments provided by the administrative software on the SUN and on other workstations.

18. TCP/IP IMPLEMENTATION SUPPORT

Research Staff:
Joel Goldberger
David Bilkis
Dale Chase
Dale Russell

Support Staff:
Manon Levenberg

18.1 PROBLEM BEING SOLVED

The Information Processing Center (IPC) supports a user community composed of internal researchers, military contractors located at bases around the country, and collaborating researchers at institutions throughout Europe. This user base is expected to expand to include researchers and military contractors in the Far East. Collaboration and communication among this distributed user base are made possible by robust and reliable network connectivity. Many of the remote sites being supported by the IPC are installing workstation environments consisting of personal workstations on local networks with gateways to the backbone Internet (ARPANET and MILNET). The TCP/IP protocol suite was specifically designed to support this connectivity and to conceal the details of the interconnection of these networks from the users.

18.2 GOALS AND APPROACH

The principal goal of the TCP/IP Implementation Support project is to install and maintain the software required for the interoperability of the ever-growing collection of long-haul and local networks. This is a very ambitious goal, and it has been extremely successful. Much of the credit for this success belongs to the Internet Concepts project at ISI and to several groups at Bolt Beranek and Newman, Inc., for their work on providing standards and testbeds for the protocols and gateways comprising the Internet. The TCP/IP Implementation Support project has been instrumental in providing a highly loaded environment in which to test the work done by these groups. The loading is a result of the size and geographical distribution of our user community and the heavy use this community makes of the facilities provided by this network connectivity.

The role of this project is to continue to work closely with these and other implementors to improve and enhance the facilities available on the ARPA-Internet. This project also works closely with other ISI internal research groups to provide facilities for prototyping and testing new network-based capabilities, including the recent work on domain servers and multimedia mail systems. Once these prototype systems are developed and debugged, this project is responsibile for their conversion to production facilities and their continuing support and maintenance.

18.3 PROGRESS

Following the installation of a BBN coded and maintained PDP-11/23 internet Gateway connecting our local 10MBit Ethernet to the ARPANET, we began expanding the population of hosts on that network. This expansion revealed a number of deficiencies both in the gateway itself and in the TCP/IP implementations on the various hosts. Much of the available applications software for these machines (VAX/UNIX, VAX/VMS, and 68000-based systems) had never been operated in this configuration. Working both independently and with the suppliers of this software, we have solved most of these problems.

We imported software from MIT to allow the connection and use of IBM Personal Computers as autonomous Internet hosts connected via our Ethernet and gateway. Working closely with the developers at MIT, we were able to isolate and correct a number of problems in this software.

We continued our investigation of the performance problems reported by the SAC and DARPA user communities in collaboration with BBN. This involved the installation of additional monitoring facilities in the TOPS-20 operating system and the coordination of tests at all four locations.

Additional hardware and software facilities were installed and integrated to support several other ISI research activities, most notably the EGP experimentation by the Internet Concepts project. We also modified our locally developed Remote-Virtual-Disk server to improve its performance and reliability. A dedicated phone line was installed and appropriate software modifications made to allow one of our local hosts to act as an Internet gateway for establishing connectivity to the University of California at Irvine.

The reliability and functionality of mail service for our hosts, both directly connected to the ARPANET and connected through the gateway, were significantly improved.

18.4 IMPACT

In addition to providing a reliable computational resource for our distributed user base, the efforts of this project have enabled a number of new research activities to be undertaker on dedicated personal workstations. This is possible because of the ease of integrating them into the ARPA-Internet via the well-defined and proven TCP/IP protocol suite. The availability of well-defined and consistent interfaces to distributed facilities has made possible the use of distributed processors for applications including the VLSI fabrication service provided by the MOSIS project at ISI, and has greatly simplified many aspects of the operation of the IPC.

18.5 FUTURE WORK

This project will no longer be conducted as a separate entity beyond the normal software maintenance activities of the IPC. The remaining work consists of fine-tuning the various implementations of the protocol suites running under the operating systems we support within the IPC (TOPS-20, UNIX, and VMS). We will also convert many of the network-based utilities to make use of the new user interface provided by the latest release of the DEC TOPS-20 operating system.

The hardware to provide direct Ethernet connectivity for our DECSystem-20 mainframes has been further delayed and is not expected until late September or early October of 1984. The software to use this hardware has been fully debugged at Stanford University, and installation should be very easy. The connectivity provided by this hardware will benefit several internal research projects at ISI that need to transfer large amounts of data among our different processors.

19. INTERLISP

***Research Staff*:**
Raymond Bates
Mark Feber

19.1 PROBLEM BEING SOLVED

This project has created a self-sustaining support center for Interlisp. The project currently provides an implementation of Interlisp, known as ISI-Interlisp, running on the VAX family of computers. It runs, in native mode, under both the UNIX (specifically Berkeley VM/UNIX, versions 4.1 and 4.2) and VMS (versions 3.0 through 3.6) operating systems.

The VAX computer was chosen as the initial machine for Interlisp implementation on the basis of its expected widespread use throughout the academic community and the opportunity for technology transfer such widespread use affords. Currently, with the expanding use of personal machines in the research community, alternate, low-cost architectures are prime candidates for the expansion of ISI-Interlisp's user base.

The emphasis of this project is to provide large-address-space portable versions of Interlisp for the VAX and for other hardware architectures.

19.2 GOALS AND APPROACH

The areas of emphasis and the goals of the Interlisp project have evolved over time. From the beginning of the project, the basic goals have been as follows:

- to support a large virtual address space;
- to be compatible with and functionally equivalent to the other major implementations of Interlisp;
- to serve as a basis for future Interlisp implementations on other mainframe computers (i.e., developing a *portable* implementation);
- to achieve reasonable performance within the limitations of the selected hardware.

However, in the project's initial phase, performance considerations were delayed in favor of speed of implementation. Additionally, some of the constraints imposed by a desire for a portable implementation were temporarily sacrificed in favor of securing the initial implementation. After the initial release, more attention was paid to performance, portability, and compatibility issues. As the project matures, these goals continue to guide the project. In its current phase, portability issues should be paramount, as the first ports to radically different architectures are anticipated. However, continued development is expected on the other fronts, since many aspects of performance can be improved and the other major Interlisp implementations continue to evolve.

19.3 PROGRESS

Notable accomplishments this year include the development of a 4.2 bsd UNIX version of ISI-Interlisp. This involved considerable effort due to the large number of critical features that were altered in the operating system between 4.1 and 4.2 bsd UNIX. In addition, two major Interlisp systems, AGE and EMYCIN, were ported to ISI-Interlisp by ISI to join the collection of other systems including Affirm, AP3, Hearsay-III, KL/ONE, and ROSIE. EMYCIN in particular has elicited a large response from our user community.

Work continues on performance issues. Additional augmentations are being made to the peephole optimizer, which was introduced in 1983. Several critical functions have been hand-coded in assembler, and a number of bottlenecks have been excised. A few small, but possibly deadly, compiler bugs have been eliminated. Improvements specific to the VMS version include a significant enhancement to the terminal I/O behavior, improved directory scanning, and the implementation of a direct-access mechanism to the underlying VMS system services. Investigation into alternate garbage collection strategies continues which, when implemented, will result in a significant increase in the amount of usable virtual memory.

In addition to the development effort, several major Interlisp tasks are ongoing. Interlisp is constantly growing and changing, and ISI-Interlisp must also change to keep up with improvements. The task of maintenance is another ongoing effort. User feedback and the maintenance effort allows for problems and bugs to be detected and removed.

This year, the Interlisp project published and presented a paper at the 1984 ACM Symposium on LISP and Functional Programming [2]. This paper describes our experiences and recommendations since the 1982 LISP conference [1]. We also prepared an exhaustive series of performance timings for presentation at this conference. This information was presented in conjunction with performance timings of all major dialects of LISP on a wide range of hardware, and showed ISI-Interlisp well within the target of expected performance given the speed of the hardware.

As of May 1984, a total of 314 systems had been distributed, with 156 targeted for UNIX and the remaining 158 for VMS.

19.4 IMPACT

The premise of the Interlisp project is that no one machine will (or should) dominate Interlisp usage in the future as the PDP-10 did in the past. Instead, new, more cost-effective machines will be appearing with increasing frequency. Therefore the new, large-address-space Interlisp has been constructed to be portable, and the Interlisp support center plans to migrate this system to new hardware on a periodic basis. The availability of Interlisp on a wide variety of machines will allow more people from the academic community, government, and private industry to work with this important language.

19.5 FUTURE WORK

In addition to the ongoing maintenance tasks, several development efforts are planned for next year:

- Port ISI-Interlisp to a different architecture. We expect this to be a major effort in the upcoming year. A 68000-based system is currently the primary candidate for this effort, as it meets the major requirements.
- Code a different garbage collector. Two different garbage collectors are being considered. ISI-Interlisp currently uses a stop-and-copy collector, which is acceptable

for UNIX but is wasteful for VMS. A better collector would be a mark-and-sweep garbage collector. A reference count garbage collector would probably be the best solution for both VMS and UNIX. but this may be more difficult to implement.

- Make ISI-Interlisp respond more effectively when a critical resource of the operating system is lacking.
- Maximize the sharing of memory among users on VMS and UNIX 4.2. The current system does not share pages among ISI-Interlisp users. By having ISI-Interlisp share pages, the operating system will be able to support more ISI-Interlisp users.
- Perform experiments in micro-coding on the VAX-11/780. A few critical functions of ISI-Interlisp will be micro-coded. We expect this to improve the speed of ISI-Interlisp substantially by removing some bottlenecks.

REFERENCES

1. Bates, R. L., D. Dyer. and J. A. G. M. Koomen, "Implementation of Interlisp on the VAX," in *Conference Record of the 1982 ACM Symposium on LISP and Functional Programming*, pp. 81-87, Association for Computing Machinery. Pittsburgh, August 1982. Also available as USC/Information Sciences Institute, RS-84-129, May 1984.

2. Bates, R. L., D. Dyer, and M. Feber, "Recent developments in ISI-Interlisp," in *Proceedings of the 1984 ACM Symposium on LISP and Functional Programming*, Association for Computing Machinery, August 1984. Also available as USC/Information Sciences Institute, RS-84-131, August 1984.

20. NEW COMPUTING ENVIRONMENT

Research Staff:	*Support Staff*:
Joel Goldberger	Manon Levenberg
Gary McGreal	
Michael Butler	
Jim Koda	
Jon Postel	

20.1 PROBLEM BEING SOLVED

For the last decade, the computational needs of researchers in all areas of computer science have been provided on large-scale timesharing machines. However, this is no longer a viable approach. ISI's two DEC mainframes running TOPS-20 are being used to full capacity. They are not capable of supporting many of the current and proposed research efforts at ISI.

As techniques and expectations (especially in the area of artificial intelligence) advance, general-purpose machines are no longer able to provide the style of interaction and capabilities — in terms of throughput and address space — required to support current research efforts. The NCE project proposes to

- provide a significant improvement in both the amount and the quality of available computing resources to the ongoing ISI research efforts through the use of dedicated personal workstations and higher capacity centralized processors and servers;
- provide for diverse access methods to the common set of services:
- fully support the use of special-purpose workstations and processors as required by individual research projects;
- free up capacity on the existing mainframes by offloading some common functions to central servers.

Due to the diversity of projects and cultures within the ISI research community, we have decided to implement a heterogeneous environment consisting of a variety of personal workstations, dedicated high-end minicomputers, and continuing use of our existing timesharing systems. Initially we will support this environment by means of autonomous, but well-coordinated, facilities for file, mail, and document manipulation (i.e., as processes on a single server providing a unified view to the outside world). This is the most expeditious way of making the existing workstations, and those we will be acquiring shortly, fully functional. In parallel, we will begin to integrate these heterogeneous systems using the DARPA/DoD protocol suite (IP, TCP, FTP, etc.), and will further provide mechanisms to promote the use of facilities among the various environments. All of the services and facilities we provide will eventually be accessible via the IP protocol suite and thus available from anywhere in the Internet.

20.2 GOALS AND APPROACH

The New Computing Environment is envisioned to consist of local processing nodes (workstations and multiuser mainframes) connected via a local network to a set of central servers. A certain minimum set of functions must be available to all nodes in this environment: mail, file manipulation (access and transfer), and Internet connectivity (Telnet). The environment provided by these nodes and servers should conceal the fine-grain detail from outside users; users external to the environment should not know what workstation a particular user is on. unless they want to. At minimum, these services should be available via any of several communication media: Local Network. Internet, and dial-up access. The NCE will provide additional support as needed to allow low-end workstations and terminals on our mainframes to utilize these services as easily as the fully capable workstations in the SUN, Dandelion, and Apollo class. Primary access to these services will be via the DoD IP/TCP protocol suite. with some extensions and enhancements to support particular modes of interaction which are not yet supported (e.g., random file access and remote-procedure calls).

The use of these services by our existing mainframes should realize immediate improvements for all users as some CPU-intensive activities are offloaded to central servers. Making these services available from our mainframes will also provide for their use from home terminals, which initially will be glass-tty's rather than autonomous processors. Access for these devices will be via dial-up lines to our existing mainframes. or to the servers themselves.

20.2.1 Overview of the Services

The following services will be provided to all nodes in the environment:

- centralized mail service,
- centralized file service,
- document formatting,
- high-quality printing,
- communications,
- special-purpose processing.

While the principal incentive for the New Computing Environment is the support and integration of a diverse set of dedicated personal workstations, not everyone at ISI requires this kind of hardware. Until they have an actual need to migrate their research efforts to workstations, a significant population will be content with our existing mainframes (KLs and VAXes). However, the enhanced facilities offered by the provision of centralized servers should also benefit this population. These enhancements will take the form of increased accessibility of files and mail, increased reliability in terms of both data integrity and security, and improved performance of the existing mainframes due to the offloading of CPU-intensive activities to dedicated servers.

20.3 PROGRESS

During this period we installed ten VAX-11/750s on the 10 MBit Ethernet and refined a remote virtual disk (RVD) protocol developed at MIT. These machines, which run with a very small local disk were made available to the local research projects as single-user Interlisp processors. and have removed a considerable load from our TOPS-20 systems. As this system moved from initial development to full production. various performance improvements were made and returned to MIT for incorporation into their later releases of the RVD software.

We installed six Xerox Dandelion workstations and their associated file and print servers. in order to begin development of the DoD standard TCP/IP protocol suite. This development has been severely hampered by Xerox internal policies. which have only recently begun to improve. We have succeeded in developing the low-level portions of these protocols and are now proceeding with the implementation of the higher levels. We also developed sophisticated network monitoring tools within the MESA environment to facilitate the isolation of performance problems on our other network-based processors.

Work also progressed on improving the support for several other models of workstations in use by the local researchers. These included the Xerox Dolphin and the Symbolics 3600 Lisp Machine. Software was installed on our existing mainframes to provide file and mail service for these workstations.

20.4 IMPACT

Our experience with personal workstations, however limited, has made it clear that they represent a useful alternative to large, timeshared mainframes for certain research activities. Several projects within the institute have already moved the majority of their work to workstations, and have therefore been able to continue research which exceeded the capacity of our mainframes. The timely support provided for different workstation models has increased the acceptance of a distributed model of computation.

Once the reliability of added facilities and of the entire environment is proven, we will offer the same type of environment to outside contractors via the Exportable Workstation Systems project. These two projects combine to provide an ideal environment for developing and refining the facilities and capabilities that are becoming increasingly important in the command and control context. These capabilities include the handling of redundant databases and the interoperability of a diverse collection of hardware.

20.5 FUTURE WORK

There is insufficient agreement among ISI project leaders on workstation hardware and software selection to develop detailed implementation plans for a user environment. A number of workstations are likely to become more widely used, while others that have been purchased will no longer be explicitly supported. All of these workstations will be enhanced by the provision of the centralized services described in the preceding section. This is the area where we will concentrate our development efforts. While most of the workstations under consideration have existing facilities to interact with remote services, they do not all use the same protocols or provide precisely the same functions.

For the near term, we will continue to support a number of incompatible server implementations running as separate processes on a single server machine. We will enhance these implementations to provide a fuller range of capabilities for immediate use by our existing workstation population. Our longer range plans call for the development of IP/TCP-based protocols to support these services. It is our intention that these services will be used by any machine included in the NCE.

20.5.1 Near-Term Goals

Our immediate goals are as follows:

- install all the servers on a single system,
- implement mail system interfaces for all workstations, and
- provide for encryption of passwords on local networks.

Each of our current workstations has its own implementation of remote file service which we have installed on one of several VAXes. These have included Symbolics 3600s, Xerox Dolphins and Dandelions, SMI SUN workstations, IBM-PCs, and SGI IRISes. These services are currently running on three different VAXes under three different versions of UNIX. All of these VAXes also support the full suite of IP/TCP protocols providing file transfer, virtual terminal, and mail connectivity with the rest of the Internet. This allows for the sharing of files and data, albeit in a rather tedious manner. Our immediate plans call for the consolidation of these services on a single VAX equipped with large-capacity disks and tape subsystems for backup. The grouping of these services on a single machine will make the exchange of files among the different environments much easier and will also improve system-level support capability.

Once all of these servers have been installed on a single machine, we will begin enhancing them to provide a wider range of capabilities, including convenient mechanisms for interfacing with the mail system of the server machine from each environment. With these enhancements in place, the present services should enable our in-house researchers to move all of their activity from our existing TOPS-20 systems to personal workstations, even before we develop a unified set of protocols and access schemes. Some of these enhancements represent significant software efforts, but they are still well below the level of effort required to implement both the user and server sides of an entirely new family of protocols.

20.5.1.1 Consolidation of support for existing servers

The following servers are presently supported:

- Chaos software for the Symbolics 3600s,
- a PUP/Leaf server for both the Dolphins and Dandelions,
- a remote disk service for VAX-11/750s and IBM-PCs,
- a different disk server for the SMI SUNs, and
- an XNS server for the SGI IRISes.

We have already begun converting these various servers to run under the latest version of UNIX (Version 4.2), but this effort is not complete. Implementation details must still be decided for the SUN virtual disk server, such as determining whether the server runs as part of the kernel or as a user process. We only recently obtained the relevant sources from SUN, so these questions have not yet been resolved.

Much of this work can and will proceed in parallel; the conversion to UNIX 4.2 is expected within one month of the time we acquire the various components.

20.5.1.2 Preliminary mail system implementation

The Information Management project has modified the Symbolics software to interface with the particular version of UNIX mail we have selected for local use. This has been integrated into their working system, and is expected to port easily to the newer version of UNIX. We have included one week in the estimate for the Chaos software conversion above, to cover any unexpected problems that may arise. Mail service for the other workstations is in various stages of development. The SUNs, which have a complete mail system, function as autonomous hosts in the Internet. We intend to rework this system to allow them to access mail held on a central server.

The IBM-PC has no mail handling system, and the Xerox Dandelions are presently equipped to interact with only a Xerox mail server. To provide mail processing facilities for these two workstations, we plan to develop a "post office" model of the existing DoD mail system (SMTP), in which an Internet host can connect (via TCP) to a "well-known" port and request the transfer of its mail. This is a relatively minor enhancement to our existing mail software, but it represents a significant step towards our longer range goal of fully centralized mail handling. To accomplish this, we will build on the IBM-PC IP/TCP implementation obtained from MIT and the MESA IP/TCP implementation developed within ISI.

20.5.1.3 Password encryption on the local networks

We have a monitoring tool, running on a Xerox Dandelion, which allows all traffic on the local Ethernet to be displayed in a number of different formats, including ASCII. This has indicated the need for security on the local net. We intend to apply encryption to all exchanges of passwords to minimize the chances that an automated password monitoring system could be installed on our local systems. We will limit this to FTP interactions initially, because the well-defined syntax of these exchanges makes them easy targets for automated monitoring. This will require changes in the VAX/UNIX, SUN, and TOPS-20 versions of FTP, in both user and server processes. A likely extension of this is to automate login on selected hosts, so that users may connect within the local environment more conveniently.

20.5.2 Long-Range Goals

Our longer range goals consist of the following:

- implementation of IP/TCP-based server access protocols,
- installation of alternative file storage technologies (laser disks, etc.),
- development of remote procedure call facilities, and
- development of dynamic resource allocation mechanisms.

20.5.2.1 Implementation of IP/TCP-based mail and file protocols

We have preliminary definitions of IP/TCP-based file and mail access protocols. These were developed from our studies of existing implementations. We will continue to refine these definitions, possibly issuing them to the Internet research community. We will then build prototype implementations on one of the existing VAX-11/750s, to further verify their functionality and completeness. Once we have a functional implementation of these protocols, we will begin implementing the client side software on the installed workstations.

20.5.2.2 Alternative file storage technologies

We have begun exploring the availability and functionality of laser disk storage systems, and we are investigating the requirements for integrating them into our environment. Assuming that these investigations yield encouraging results, we will begin the design of both the hardware interfaces and the supporting software to use these devices. This development will take place on one of our existing VAX-11/750s. The initial testing will be conducted using the prototype file access software mentioned in the section above.

20.5.2.3 Remote procedure call and dynamic resource allocation facilities

Remote procedure call facilities and dynamic resource allocation techniques are areas of proposed research in the Internet project, and we intend to build on their results as they become available. This effort is expected to continue over the next two to three years, with early prototypes available in late 1985.

21. STRATEGIC COMPUTING - DEVELOPMENT SYSTEMS

Research Staff:
Gary McGreal
Dan Pederson

21.1 PROBLEM BEING SOLVED

Utilizing recent advances in artificial intelligence. computer science. and microelectronics. DARPA plans to create a new generation of "machine intelligence technology." The DARPA Strategic Computing program will be targeting key areas where broad advances can be made in machine intelligence technology and applications of the technology to critical problems in defense can be demonstrated.

The Strategic Computing program will be supported by a technology infrastructure, which will provide current state-of-the-art computing technology. network communications, and shared resources to the selected program participants.

This project will provide development computers for the Strategic Computing program, system integration as required, and distribution mechanisms for disseminating the systems to the program participants. In addition, the project will define an architecture for system communications and resource sharing among the computers acquired.

21.2 GOALS AND APPROACH

A number of machines have been developed to support high-speed processing of large symbolic programs. Each of these machines provides an extensive interactive programming environment, a sophisticated display manager "window system," a real-time window-oriented editor, incremental compilers. and dynamic linking. These systems are the state of the art in program development environments. They are widely used in the research community for systems development, expert systems, natural language systems, and mapping applications, and for supporting CAD/CAM environments.

Because these systems are very expensive, it is important to use them as efficiently as possible. Since they are single-user systems, they can not be timeshared in the traditional sense. The machines are too expensive to be supplied on a one-to-one basis to researchers; thus workers are currently placed in the awkward situation of having to schedule their computing needs. The resultant scheduling conflicts naturally lead to low researcher productivity.

An examination of dynamic machine utilization shows that many activities do not require the high-speed processing capabilities afforded by these machines. Slower, less expensive. general-purpose machines supporting the same language base are becoming available. These would be adequate for many of the less intensive research activities such as editing, text preparation, and file scanning. They are too slow. however. for more intensive program development and execution activities.

The different resources needed for different types of activities and the availability of machines of appropriate cost/performance for carrying out those activities suggest a workstation/server architecture based on these two classes of machines and an interconnecting IP/TCP-based network. In order to support dynamic source code exchange between the server and the workstation, it will be necessary for them to support the same AI language system. Common LISP is the natural choice in such an architecture, as it was designed with the concepts of commonality and portability as primary goals.

A significant benefit of such an architecture is the ability of researchers to communicate over the Internet to utilize high-powered resources available on prototype and low-production machines. This will be particularly useful as prototype machines are developed under the machine architectures phase of the program.

ISI proposes to acquire a mix of these machines to support the requirements of the Strategic Computing program, and to integrate and test individual systems as required for particular Strategic Computing program participants. The integration tasks will avoid duplication of effort among Strategic Computing developers. ISI will collect software to support this architecture as it is developed by vendors and by the research community. ISI will test, modify, and augment software as required; provide support for software exchange and distribution among DARPA development sites; assure working compatibility of the Common LISP systems; work with commercial vendors to resolve vendor-specific problems; and work to allow network compatibility of the systems with DoD network protocols (IP/TCP). The software and system integration efforts will be carried out at the ISI Marina del Rey facility.

21.3 PROGRESS

An initial order for LISP machines was made during this reporting period. The machines acquired are Symbolics 3600 systems with 4 MB main memory and 474 MB local disk. A local cluster was installed at ISI to test the hardware and software. Problems were found in the network software and the displays. These problems were resolved with assistance from the vendor.

Currently, eight machines have been sent out for preliminary work in the Strategic Computing program. These machines have gone to the National Security Agency, the BDM Corporation, and the University of Maryland. In addition, several machines are in use at ISI in preliminary work on natural language and the Common LISP framework effort.

21.4 IMPACT

The DARPA Strategic Computing program will develop and integrate advanced computer technology into military applications. Technological advancement will be sought in the areas of microelectronics and high-speed symbolic machines, as well as the application of this technology to the military. Potential military applications of this technology include autonomous systems, battlefield management and assessment, planning and advising, and simulation systems.

The initial program applications include an autonomous vehicle, a pilot's associate, and a battle management system. These applications test the evolving technology in the areas of vision, expert systems, speech recognition, and high-performance knowledge processing systems. Each application seeks to demonstrate the new technologies' potential for providing a major increase in defense capability and to reflect a broad demonstration base that is relevant to each of the three services. The development systems acquired by this project will support the technology base and the targeted applications under the Strategic Computing program.

21.5 FUTURE WORK

During the coming year, ISI will negotiate with qualified vendors and acquire a mix of high-end LISP machines and workstations. As the workstation/server architecture is defined, ISI will configure systems, test vendor software, and distribute systems to Strategic Computing program participants as required and directed by DARPA.

22. STRATEGIC C3 SYSTEM EXPERIMENT SUPPORT

***Research Staff*:**
Stephen Dougherty
Victor Ramos

22.1 PROBLEM BEING SOLVED

DARPA has defined an experiment in Strategic C3 systems to be conducted in cooperation with the *World Wide Military Command Control System* (WWMCCS) System Engineer (WSE) and the Strategic Air Command (SAC). The concept of the experiment is to demonstrate and evaluate the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-base techniques) for strategic command, control, and communication.

ISI's portion of the plan is to provide an initial core of necessary facilities (ARPANET/MILNET access, host systems, various software tools, Network Services support, etc.) to allow SAC personnel to gain experience with this technology and to ensure the success of the experiment. Specifically, SAC must have fairly broad-based experience with ARPANET-based on-line interactive computing. Installation and maintenance of modems and of 55 or more interactive CRT terminals, user training, system software training and support, and on-site maintenance of equipment are part of the continuing program.

22.2 GOALS AND APPROACH

The total specific goals of this DARPA experiment are to:

- Demonstrate and evaluate the survivability of multinode computer-communication networks, including the ARPANET and packet radio, especially for remote access from both airborne platforms and surface sites.
- Explore replication and reconstitution of critical knowledge bases on this network in the face of a loss of a large number of links.
- Demonstrate and evaluate the rapid reconstitution of a network by rapid deployment of packet radio nets to reestablish connectivity between surviving elements of the network.
- Conduct experiments and exercises to evaluate the utility of such a network on a distributed knowledge base to support postattack C3 activities.

The experiment is defined as having three phases:

1. Phase I is planned to demonstrate air-to-surface packet radio links and gateways into the ARPANET as a first step in evaluating the feasibility of a truly survivable strategic network.
2. Phase II is directed toward creating a survivable message system and databases through multiple copies of the critical components and data across the ARPANET.
3. Phase III will address the feasibility of rapid reconstitution of a strategic network by deployment of packet radio networks to reconnect surviving elements of the network.

22.3 PROGRESS

The major activity at SAC during this period (outside of the normal datacom. experiment, and Network Services support) revolved around the ARPANET/MILNET split. The SAC user community has been growing regularly since its inception. As a result. existing resources have been strained. An additional Terminal Access Controller (TAC) was installed to support the growing user community and to facilitate the upcoming split. The dedication of ISIE to SAC helped eliminate user frustration and increase productivity due to the additional computational capacity available.

A SAC data communications reorganization plan, developed by ISI and proposed in May 1983, increased the maximum number of SAC users able to access the ARPANET/MILNET and substantially improved the maintainability of the data communications resources. The plan was accepted and the work was completed in August 1984.

In addition, new resources were obtained or redistributed to support the VAX van effort (the effort is a portion of the experiment testing the mobility and reconstitutability of a partially destroyed network). Additional ARPANET/MILNET access was provided to SRI International (which was working the Telecommunications portion of this effort and housed the vans) via 9600 bps communications lines. A plan for reallocation of C3 terminal assets, identified for use on the VAX van by the Headquarters Emergency Relocation Team (HERT) participants, was approved by the Chief of Staff (Monroe W. Hatch) and the Deputy Chief of Staff for AD (James L. Crouch). ISI also supplied new printers for the vans and is helping in the coordination and allocatinn of data communications resources (both 9600 and 56000 bps) to the van effort.

22.4 IMPACT

One of the premises of this experiment is that SAC personnel will become more proficient and knowledgeable users of the current computer technology available within the ARPANET/MILNET arena. This knowledge will allow SAC to make more effective evaluations of new technologies for strategic use. to identify areas of potential future research. and to implement those technologies found appropriate.

22.5 FUTURE WORK

ISI will continue to assist DARPA in planning this program, working to satisfy the communication and computer resource requirements of SAC headquarters. In particular. ISI will do the following:

- Continue to provide on-site maintenance support for the required equipment.
- Continue to plan and assist in implementing improved SAC connectivity to the MILNET/DDN/ARPANET.
- Install and maintain terminals and communication equipment for the connection of several Air Force bases to the MILNET. to allow increased participation in the experiment.
- Continue to supply computational. programming. and administrative support to users at SAC headquarters via the resources of the USC-ISIE computer. the programming, operations. and Network Services staff in Marina Del Rey. and the on-site technician.

ISI's most visible direct support will continue to be the on-site technician at SAC. who will be responsible for the identification of system malfunctions and for primary maintenance of on-site equipment He will be supplied with required spare parts and will have maintenance contracts with the equipment vendors. Further support will be available from ISI in terms of additional spare parts. systems expertise, and documentation as necessary. The on-site maintenance technician will also be responsible for the off-site terminals at Vandenberg Air Force Base. Barksdale Air Force Base. March Air Force Base, and new locations as dictated by the requirements of the experiment. He will coordinate data communications requests of SAC AD with SRI International and SAC under the supervision of ISI management. He will also provide training and consulting to SAC personnel with backup from ISI as required.

ISI will provide program planning assistance to DARPA. We will continue to investigate the data communications requirements for SAC Headquarters. the HERT effort. and the ACCESS system (SAC Automated Command and Control Executive Support System). As specific research tasks are defined. ISI may choose to submit new proposals to address one or more of these tasks.

23. PUBLICATIONS

Books, Chapters, and Journal Articles

1. Mann, W. C.. "Systemic encounters with computation." *Network: News, Views and Reviews in Systemic Linguistics and Related Areas*, (5), May 1983, 27-33.

2. Matthiessen, C. M. I. M., Choosing Tense in English, Master's thesis, University of California at Los Angeles, 1983.

3. Matthiessen, C. M. I. M., "Choosing primary tense in English," *Studies in Language* 7, (3), 1983.

4. Matthiessen, C. M. I. M., "What's in Nigel: 1," *Network: News, Views and Reviews in Systemic Linguistics and Related Areas*, (6), April 1984, 36–44.

5. Mostow, J., "1983 International Machine Learning Workshop: An informal report," *SIGART Newsletter*, (86), October 1983, 24–31.

6. Mostow, J., and R. M. Balzer, "Application of a transformational software development methodology to VLSI design," *Journal of Systems and Software*, (4), 1984.

7. Mostow, J., "A decision-based framework for comparing hardware compilers," *Journal of Systems and Software*, (4), 1984.

8. Resnick, L. B., and R. Neches, "Factors affecting individual differences in learning ability," in R.J. Sternberg (ed.), *Advances in the Psychology of Human Intelligence*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

9. Reynolds, J., and J. Postel, "Domain style names adopted by internet," *CSNET News*, (4), Spring 1984, 1-5. Published by CSNET CIC.

10. Swartout, W. R., "XPLAIN: A system for creating and explaining expert consulting programs," *Artificial Intelligence Journal* 21, (3), September 1983, 285–325. Also available as USC/Information Sciences Institute, RS–83–4, July 1983.

11. Weischedel, R. M., and N. K. Sondheimer, "Meta-rules as a basis for processing ill-formed input," *American Journal of Computational Linguistics* 9, (3–4), 1983. Also appears as USC/Information Sciences Institute, RS–84–146.

12. Wile, D. S., "Program developments: Formal explanations of implementations," *Communications of the ACM* 26, (11), 1983, 902–911.

Refereed Conference Proceedings and Papers

1. Balzer, R. M., D. Dyer, M. Morgenstern, and R. Neches, "Specification-based computing environments," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Washington, D. C., 1983.

2. Barnett, J. A., and Donald Cohen, "A wrinkle on satisficing search problems," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 774–776, Karlsruhe, Germany, August 1983.

3. Barnett, J. A., "Optimal searches from AND and OR nodes," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 786–788, Karlsruhe, Germany, August 1983.

4. Bates, R. L., D. Dyer, and J. A. G. M. Koomen, "Implementation of Interlisp on the VAX," in *Conference Record of the 1982 ACM Symposium on LISP and Functional Programming*, pp. 81–87, Association for Computing Machinery, Pittsburgh, August 1982. Also available as USC/Information Sciences Institute, RS–84–129, May 1984.

5. Casner, S. L., Danny Cohen, and E. R. Cole, "Issues in satellite packet video communication," in *Conference Record of the International Conference on Communications*, IEEE, Boston, June 1983. Also available as USC/Information Sciences Institute, RS–83–5, July 1983.

6. Cohen, Danny, "Real-time packet video over satellite channels," in *Proceedings of the Eighth Data Communications Symposium*, pp. 40–47, ACM/IEEE, Falmouth, Massachusetts, October 1983. Also available as USC/Information Sciences Institute, RS–83–118, March 1984.

7. Cohen, Danny, and L. Johnsson, "Mathematical approach to computational networks," in *Proceedings of the IEEE International Conference on Computer Design (ICCD–83)*, pp. 642–646, IEEE, October 1983.

8. Cohen, Danny, and J. B. Postel, "Gateways, bridges, and tunnels in computer mail," in *Local Net 83 – Strategy and Systems*, London, Online Conferences, March 1983. Also in *Local Net 83 – Distributed Office and Factory Systems*, New York, Online Conferences, September 1983. Also available as USC/Information Sciences Institute, RS–83–117, January 1984.

9. Cohen, Danny, and J. B. Postel, "The ISO reference model and other protocol architectures," in R. E. A. Mason (ed.), *Information Processing '83*, IFIP, Paris, September 1983. Also available as USC/Information Sciences Institute, RS–83–6, November 1983.

10. Cohen, Danny, "The MOSIS story," in *The 4th Jerusalem Conference on Information Technology*, pp. 650–657, IEEE, Jerusalem, May 1984. Also available as USC/Information Sciences Institute, RS–84–139, July 1984.

11. Cohen, Danny, "Satellite communication of real-time packet video images," in *Proceedings of the International Conference for Computer Communications*, International Council for Computer Communications, Sydney, Australia, 1984. Also available as USC/Information Sciences Institute, RS–84–136, June 1984.

12. Cohen, Donald, "Symbolic execution of the Gist specification language," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 17–20, Karlsruhe, Germany, August 1983.

13. Feather, M. S., "Reuse in the context of a transformation based methodology," in *Proceedings of the Workshop on Reusability in Programming*, pp. 50–58, ITT Programming, Newport, Rhode Island, September 1983. Also available as USC/Information Sciences Institute, RS–83–125, April 1984.

14. Feather, M. S., "Specification and transformation: Automated implementation," in P. Pepper (ed.), *NATO ASI Series F: Computer and Systems Sciences.* Volume 8: *Program Transformation and Programming Environments: Report on a Workshop Directed by F.L. Bauer and H. Remus*, pp. 223–230, Springer-Verlag, 1984. Also available as USC/Information Sciences Institute, RS–83–124, April 1984.

15. Goldman, N. M., "Three dimensions of design development," in *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, Washington, D. C., August 1983. Also available as USC/Information Sciences Institute, RS–83–2, July 1983.

16. Kaczmarek, T., W. Mark, and N. Sondheimer, "The Consul/CUE interface: An integrated interactive environment," in *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 98-102, ACM/SIGCHI, December 1983.

17. Kaczmarek, T., and N. Sondheimer, "Man-machine interface modes: Variety and integration," in *Proceeedings of the 1984 Conference on Intelligent Systems and Machines*, Rochester, Michigan, April 1984.

18. Katz, A., "An experimental internetwork multimedia mail system," in *Proceedings of the IFIP 6.5 Working Conference on Computer Message Services*, IFIP, Nottingham, England, June 1984. Also available as USC/Information Sciences Institute, RS-84-134

19. Lewicki, G., Danny Cohen, P. Losleben, and D. Trotter, "MOSIS: Present and future," in *Proceedings of the MIT Conference on Advanced Research in VLSI*, pp. 124-128, MIT, Cambridge, Massachusetts, January 1984. Also available as USC/Information Sciences Institute, RS-83-122, March 1984.

20. Mann, W. C., "An overview of the Penman text generation system," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 261-265, AAAI, August 1983. Also appears as USC/Information Sciences Institute, RR-83-114.

21. Mann, W. C., "Inquiry semantics: A functional semantics of natural language grammar," in *Proceedings of the First Annual Conference of the European Chapter of the Association for Computational Linguistics*, ACL, September 1983. Also available as USC/Information Sciences Institute, RS-83-8, October 1983.

22. Mann, W. C., "A linguistic overview of the Nigel text generation grammar," in *The Tenth LACUS Forum*, Hornbeam Press, August 1983. Also available as USC/Information Sciences Institute, RS-83-9, October 1983.

23. Matthiessen, C. M. I. M., "How to make grammatical choices in text generation," in *The Tenth LACUS Forum*, Hornbeam Press, 1983. Also available as USC/Information Sciences Institute, RS-83-120, February 1984.

24. Matthiessen, C. M. I. M., "Systemic grammar in computation: The Nigel case," in *Proceedings of the First Annual Conference of the European Chapter of the Association for Computational Linguistics*, ACL, September 1983. Also available as USC/Information Sciences Institute, RS-83-121, February 1984.

25. Mockapetris, P., "Analysis of reliable multicast algorithms for local networks," in *Proceedings of the Eighth Data Communications Symposium*, ACM/IEEE, Falmouth, Massachusetts, October 1983. Also available as USC/Information Sciences Institute, RS-83-10, November 1983.

26. Mockapetris, P., J. Postel, and P. Kirton, "Name server design for distributed systems," in *Proceedings of the Seventh International Conference on Computer Communication*, ICCC, Sidney, Australia, October 30 to November 3 1984. Also available as USC/Information Sciences Institute, RS-84-132, June 1984.

27. Mockapetris, P., "The domain name system," in *Proceedings of the IFIP 6.5 Working Conference on Computer Message Services*, IFIP, Nottingham, England, May 1984. Also available as USC/Information Sciences Institute, RS-84-133, June 1984.

28. Morgenstern, M., "Active databases as a paradigm for enhanced computing environments," in *Proceedings of the Ninth International Conference on Very Large Data Bases*, October 1983. Also available as USC/Information Sciences Institute, RS-83-7, October 1983.

29. Morgenstern, M., "Constraint equations: A concise compilable representation for quantified constraints in semantic networks," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Washington, D. C., August 1984. Also available as USC/Information Sciences Institute, RS-84-137, June 1984.

30. Mostow, J.. "Program transformations for VLSI," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 40-43, Karlsruhe, Germany. August 1983.

31. Mostow, J., "A problem-solver for making advice operational," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Washington, D. C., August 1983.

32. Neches, R., R. M. Balzer, D. Dyer, N. M. Goldman, and M. Morgenstern, "Information Management: A specification-oriented, rule-based approach to friendly computing environments," in *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, IEEE, Bombay, India, 1983.

33. Schmolze, J., and T. Lipkis, "Classification in the KL-ONE knowledge representation system," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, 1983.

34. Swartout, W. R., "The GIST behavior explainer," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Washington, D.C., August 1983. Also available as USC/Information Sciences Institute, RS-83-3, July 1983.

35. Wile. D. S., "Transformation-based software development," in P. Pepper (ed.). *NATO ASI Series F: Computer and Systems Sciences. Volume 8: Program Transformation and Programming Environments: Report on a Workshop directed by F.L. Bauer and H. Remus*, pp. 323-330, Springer-Verlag, 1984.

Technical Reports

1. Bisbey, R., II, D. Hollingworth, and B. Britt, *Graphics Language (Version 2.2)*, USC/Information Sciences Institute, TM-80-18.1, May 1984.

2. Mann, W. C., *Discourse Structures for Text Generation*, USC/Information Sciences Institute, RR-84-127, February 1984. Also to appear in the proceedings of the 1984 COLING/ACL conference, July 1984.

3. Mann, W. C., and S. A. Thompson, *Relational Propositions in Discourse*, USC/Information Sciences Institute, RR-83-115, July 1983. To appear in *Discourse Processes*.

4. Moses, L., and S. Coyazo, *Using Scribe to Select Fonts on the Penguin*, USC/Information Sciences Institute, TM-83-119, February 1984.

5. MOSIS Project, *The MOSIS System (what it is and how to use it)*, USC/Information Sciences Institute, TM-84-128, March 1984.

6. Robinson, J. J., *Extending Grammars to New Domains*. USC/Information Sciences Institute, RR-83-123, January 1984.

Informal Project Notes

1. DeSchon, A.. "Facsimile Support Programs User Guide." USC/Information Sciences Institute, September 1983.
2. Hinden, R., J. Postel, M. Muuss, and J. Reynolds, "Gateway Special Interest Group Meeting Notes." USC/Information Sciences Institute. RFC 898. April 1984.
3. ISI Network Services Group, "NCP Calc User's Guide, Version 1.712A." 1983.
4. ISI Network Services Group, "User's Guide to TOPS-20." April 1984.
5. ISI Network Services Group, "New User's Guide to VAX/VMS." May 1984.
6. Kaczmarek, T., "CUE Forms Description," CUE Working Paper, 1984.
7. Katz, A., "MMM – A Multimedia Mail System User Interface," USC/Information Sciences Institute, July 1983.
8. Mockapetris, P., "Domain Names – Concepts and Facilities," USC/Information Sciences Institute, RFC 882, November 1983.
9. Mockapetris, P., "Domain Names – Implementation and Specification," USC/Information Sciences Institute, RFC 883, November 1983.
10. Postel, J., "The TCP Maximum Segment Size and Related Topics," USC/Information Sciences Institute, RFC 879, November 1983.
11. Postel, J., "The Domain Names Plan and Schedule," USC/Information Sciences Institute, RFC 881, November 1983.
12. Postel, J., "Telnet End of Record Option," USC/Information Sciences Institute, RFC 885, December 1983.
13. Postel, J., "Exterior Gateway Protocol Implementation Schedule," USC/Information Sciences Institute, RFC 890, February 1984.
14. Postel, J., "Domain Name System Implementation Schedule," USC/Information Sciences Institute, RFC 897, February 1984.
15. Postel, J., "A Standard for the Transmission of IP Datagrams Over Experimental Ethernet Networks," USC/Information Sciences Institute, RFC 895, April 1984.
16. Postel, J., and A. Westine, "Requests for Comments Summary," USC/Information Sciences Institute, RFC 899, May 1984.
17. Reynolds, J., "How to Create and Print a Bitmap Using a Hardcopy Picture," USC/Information Sciences Institute, August 1983.
18. Reynolds, J., and J. Postel, "Assigned Numbers", USC/Information Sciences Institute. RFC 870, October 1983.
19. Reynolds, J., and J. Postel, "Official Protocols," USC/Information Sciences Institute, RFC 880, October 1983.
20. Reynolds, J., "ISI-Hosts." USC/Information Sciences Institute, June 1984.

21. Reynolds, J., and J. Postel, "Assigned Numbers," USC/Information Sciences Institute, RFC 900, June 1984.

22. Reynolds, J., and J. Postel, "Official ARPA-Internet Protocols," USC/Information Sciences Institute. RFC 901, June 1984.

23. Smallberg, D. . "Survey of SMTP Implementations." USC/Information Sciences Institute. RFC 876, November 1983.

Miscellaneous Materials

1. Neches, R., The Information Management computing environment. Videotape. Shown at the CHI '83 Conference on Human Factors in Computing. December 1983.

RESEARCH AND ADMINISTRATIVE SUPPORT

Institute Administration:
Robert Blechen
Larry Ball
Monica Boseman
Patti Craig
Barbara Davey
Richard Dolen
Phyllis Keiser
Gary Lum
Karen Luna
Gina Maschmeier
Heidi Mucha
Dezra Seale
Karen Totino
Lani Upton
Rolanda Valentin

Librarian:
Alicia Drake

Publications:
Victor Brown
Sheila Coyazo

Secretary to the Director:
Barbara Brockschmidt

Computing Facilities
Training and Information
Chloe Holg
Lisa Moses

Development Laboratory
Robert Parker
Shorty Garza
Bob Hines
Jeff LaCoss
Lee Magnone
Rick Shiffman
Jerry Wills
Rob Wormuth

INFORMATION
SCIENCES
INSTITUTE

4676 Admiralty Way/Marina del Rey/California 90292-6695