A perspective on network operating systems*

by STEPHEN R. KIMBLETON and RICHARD L. MANDELL
USC/Information Sciences Institute
Marina del Rey, California

ABSTRACT

The viability of packet switched computer communication has been demonstrated. The potential for more effective computing through resource sharing and load balancing is evident. Realization of this potential requires expanded user support to reduce or eliminate much of the need for users to learn the command languages of the hosts being accessed and of the communications subnetwork. Such a capability can be provided by a mediating agent providing ease of access to resources and control of resource access—a role traditionally ascribed to an operating system in the context of an individual computer system. This mediating agent, hereafter termed a Network Operating System (NOS), requires careful exploration to determine its appropriate interaction with the operating systems of the hosts within the network. This paper discusses the functions required of a Network Operating System and identifies major differences between the role of the Network Operating System and an individual host operating system. As such, it is intended to provide a basic perspective on the field of Network Operating Systems.

INTRODUCTION

Early computing experience quickly demonstrated the undesirability of requiring the user to directly interact with and control raw physical resources. As a result, operating systems were introduced which provided two primary functions: (i) provision of ease of access to computing resources, and (ii) control of the allocation of resources across multiple competing requests. Thus, the operating system may be regarded as an agent interposed between the user and the system resources.

Heterogeneous networks, such as ARPANET,\textsuperscript{1-4} potentially permit both sharing of dissimilar resources and balancing of workload across a collection of similar resources such as the Operating System Class (OSC) consisting of the collection of TENEX operating systems within ARPANET. Presently, the viability of packet switched computer communications technology as used in ARPANET may be regarded as proved. However, the user wishing to make full use of the resource sharing potential of a heterogeneous network is currently faced with a requirement for learning the command languages of each separate system being accessed and, additionally, the command language required to support network communication. For the casual (non systems programmer) network user, this constitutes an immense startup cost which must be incurred prior to any reasonable use of network resources. The natural result is a minimization of the tendency to utilize potentially more effective remote resources.

Upon reflection, the obvious conclusion is that another entity, hereafter termed a mediator or Network Operating System (NOS), is required to interface with the collection of host operating systems. In recognition of the urgent necessity for such a capability, an increasing number of papers are undergoing gestation and are scheduled to appear. The objective of this paper is to provide a global overview of the subject; to indicate some of the major components required for an NOS; to discuss the results of an NOS study\textsuperscript{5} and to provide context for the remainder of the papers in this session.

It should be noted that the discussion of NOS in this paper is predicated upon an assumption that the existing host operating systems must remain essentially intact. In the alternative case in which one is free to design the operating systems for both the computer and the communications capabilities, substantial simplifications can result as the discussion in References\textsuperscript{6} and \textsuperscript{7} demonstrate.

To provide a perspective for reading subsequent sections, the remainder of this section identifies major differences between an NOS and an OS, establishes some of the major NOS objectives, and identifies requirements implicit in NOS design. The second section then discusses the major classes of primitives required for NOS implementation. Based upon this discussion, the two subsequent sections discuss issues and implica-
tions of the primitives required for data migration and network job execution. The fifth section examines the relationship of this work to existing capabilities and the sixth section closes with some summary remarks.

**NOS-OS similarities and differences**

From the viewpoint of an individual user, the basic atoms of both a network and an individual computer system are job steps and data. However, several major differences exist including: control of individual host resources; encapsulation; heterogeneity; geographical separation; and differing organizational constraints.

An individual operating system provides direct access to and control of individual host resources. In contrast, a Network Operating System provides a mediator which, to avoid substantial modifications to host operating systems, interfaces with the collection of operating systems to effect the requested functions. This has the advantage of eliminating the need for writing device drivers and many system utilities. The disadvantage is the need for explicit interfacing with a collection of distinct operating systems which may not provide comparable collections of capabilities.

A Network Operating System, in contrast with an individual operating system which has total control of system resources through effective encapsulation of the user, only mediates the interaction of the user with the operating system. Thus, the individual network user provided with access to a given system is assumed to have effectively the same potential spectrum of available capabilities as a local user. Moreover, since the user is not encapsulated, and the NOS is designed to support general purpose computing, it will usually be necessary to have some knowledge of local systems for interpretation of diagnostics and miscellaneous system responses created by program or programming errors. However, remote procedure calls, invocation of debugged programs, and many data operations could proceed without such knowledge.

For some applications, encapsulation may prove mandatory. Thus, the National Software Works provides such a capability in the context of providing a software production capability. This requires interrupt capturing, developing appropriate diagnostic translators and, for comparable items of software executing on distinct systems, development of standardized translators (grammars). This knowledge of the program to be executed is a prerequisite for its encapsulation. Encapsulation clearly provides significantly greater control of both user and host and, conceptually, is closer to heterogeneous multiprocessing. In view of the NSW demonstration of the feasibility of encapsulation, determination of its desirability is effectively a cost benefit exercise.

Host heterogeneity raises issues for both job processing and data movement. Job processing effectiveness is expedited through provision of a common command language. However, as discussed above, determination of the proper amount of uniformity to provide in diagnostic and control messages may be regarded as an open issue. The data implication of host heterogeneity is reflected in the need for data selection, translation and transformation as discussed later.

The intent within a Network Operating System is to provide a collection of capabilities which enables an individual user to access remote resources (programs, data or systems) just as if the user were local to all these resources. Formally, this objective, although feasible, still requires caution on the part of the user in view of the time delays induced through accessing remote resources. This issue has been discussed in the context of providing a network based programming environment and the care required seems similar to that required by the advent of virtual systems. That is, assuming that virtual memory was identical to real memory it could, for programs with poor locality, result in both poor utilization of system resources and high delay in program execution. It seems likely that improper utilization of network resources via an NOS will be reflected in the same manner.

Differing organizations have different viewpoints regarding sharing, accounting, and control of resource utilization. In a local environment these differences quickly become known and, through negotiation, become acceptable. In contrast, in a networking environment, the identity of the real user would usually be unknown and more formalized and rationalized procedures must be used. The precise implications of these organizational differences in style seem to be an open issue whose resolution is likely to be slow. However, the implications of poor accounting algorithms are already becoming manifest.

**NOS objectives and requirements**

Five primary issues must be considered in developing a network operating system:

- provision of a uniform user viewpoint of resources
- modular expansibility
- control of host network interaction
- allocation of global network resources to network computing requirements, and
- implementation mode

The first objective was discussed above. The second is required to permit hosts to provide varying levels of NOS support as dictated by need. The third is required to assure adequate protection of host interests in a networking environment. The fourth is required to support efficient network based computing in general, and is mandatory for Mission Oriented Networks in which a “collection” of computers must work cooperatively to achieve the organizational information processing function. The fifth consideration is required to ensure
that the stated objectives do, indeed, prove economically feasible.

The developer of an operating system is effectively free to start from scratch. In contrast, the developer of a Network Operating System must interface with the existing software technology. Moreover, to be truly useful, the developer must also accept the fact that different sites are likely to have differing aspirations regarding network utilization and, as a result, will be likely to support different levels of sophistication in their NOS. It follows that a realistic NOS must be modularly expandable; the evident corollary is that it must support interaction with users not wishing to participate in the NOS as would be the case in ARPA N ET for users wishing to rely upon the existing protocols.

In almost any scientific or engineering endeavor, the first objective is to show that something can be done. The second is to show that it can be done well and the third is control. Control, in an NOS context, requires control of the host network interaction to prevent network demands from unfairly impacting local users. In addition, control in the context of a Mission Oriented Network is also required to balance resource requirements against resource requests.

The mediator viewpoint described supports basic implementation and refinement of an NOS. The required element of control, however, imposes a host requirement significantly different from that currently existing since the network user is a relatively more unknown issue. As a result, the resource requirements of such a user are less predictable and the need for online, near real time control is increased. This, in turn, requires automated data gathering, archiving, and analysis. In addition, it requires the existence of a centralized network management capability to provide a common access point for users, individual installations and the network to obtain status and availability information. Such a capability would also permit balancing resource requirements against resources.

Host operating systems are complex, packet switched subnetworks are complex, and it is reasonable to assume that a completed network operating system will be complex. This necessitates careful consideration of its implementation mode. Two major alternatives exist: implementation within the host or implementation via augmentation of the host with a separate computer interposed between the host and the packet switch (IMP). In our opinion, the dramatic decline in hardware costs coupled with the increasing expense of sophisticated systems programmers argues strongly for the augmentation approach. Utilization of such an approach permits centralized design, implementation and support of the Network Operating Systems. Thus, only the host interfaces need be separately tailored. It should be noted that this support processor differs in scope from that usually subsumed under the title Front End Processor, which is primarily concerned with offloading common portions of the software required to support ARPA N ET protocols. However, both have in common the need for case by case implementation of host interfaces.

NOS PRIMITIVE CATEGORIES

The global objectives of a Network Operating System require development of primitives in four categories:

- user communication,
- data migration,
- network job execution, and
- control.

User communication

User communication primitives are required since, in a geographically dispersed environment, the communication alternatives would require utilization of telephone, telegraph, etc. However, in view of the need to transmit programs, data, documentation, and user guidance (counseling), provision of a suitable mechanism within the computer communications network is clearly required. It should be noted that some of the more sophisticated individual host operating systems also provide such a capability. Currently, rather general capabilities for user communication are available within the ARPANET for the TENVEX subcollection of hosts.

In passing, it is worth noting that although rather general capabilities already exist, substantially more sophisticated capabilities can be considered. Two major branches can be distinguished: message processing and teleconferencing.

Message processing provides five basic categories of services: creation, coordination, forwarding, alerting, and event processing. Through their provision, the ability of a large organization to respond in a timely manner to the dynamics of the environment is substantially facilitated. Creation and coordination permit generation and refinement of a proposed message by the collection of relevant originators. Forwarding encompasses the transmission process. Alerting supports early notification of the appropriate spectrum of recipients as determined from the addressee list or via content analysis. Finally, event processing can permit automatic invocation of computer operations appropriate to the message content (e.g., inventory reorder upon notification of outages in intermediate warehouses).

As observed in Reference 4, "Teleconferencing encompasses both multiparty voice/visual communication and a more formalized collection of capabilities related to hardcopy message communication. A particular form of teleconferencing, computer-based conferencing, provides a natural support basis for communication among geographically dispersed computer users. This latter form of communication is also advantageous
when it is appropriate to maintain permanent conference records, e.g., command and control. The general capabilities provided include archiving, indexing, searching, and updating of conferences. Through their utilization, communication between users is substantially enhanced."

**Data migration**

Data migration is the basic capability provided within a network for access to remote data. Current ARPAואネット provided capabilities primarily support transmission of sequential text files or block transmission of binary files. However, as discussed in the following section, efficient network utilization requires substantially more sophisticated capabilities. The spectrum of capabilities required for data migration is discussed later.

**Network job execution**

Network job execution differs from job execution in an individual computer system in the possibilities for: concurrent execution of parallel job steps; migration of a given job step to alternative sites; synchronization of job step execution across hosts; and the need for control of these capabilities. Moreover, in a heterogeneous network, job execution will generally require access to remote data. These and other issues are discussed in the next section.

**Control**

Control in a computer communications network can be considered at three levels:

- subnetwork control,
- host control, and
- network (host plus subnetwork) control.

Current subnetwork control capabilities are primarily limited to topology and link capacity modification. As a result, the control actions which can be invoked are rather static and the time constant for their invocation is on the order of months.

In the future, such static approaches to control promise to be unacceptable due to the emergent need for handling multimodal traffic including: 227

- interactive traffic
- high throughput traffic such as file transfer,
- real time traffic such as digital transmission of speech, and
- guaranteed bandwidth traffic.

Since the available capacity directly interconnecting any two packet switches is relatively static, assurance of an equitable allocation of capacity among these traffic types as well as provision of guarantees that limited amounts of traffic of each type will get through requires careful investigation. An immediate requirement for such an investigation is online, near real time monitoring of traffic conditions and a likely corollary is an ability to effect control actions in near real time. The importance of research in this area is only beginning to be perceived with the transition of packet switching communication capabilities from the status of a research instrument to the status of a utility.

In a Value Added Network, by definition, 4 individual hosts as well as the subnetwork are organizationally independent. As a result, there is little need to coordinate and control the effects of job and data assignments. In contrast, in a Mission Oriented Network, a collection of hosts as well as, perhaps, the subnetwork are expected to work together cooperatively to achieve the organizational information processing function. This, in turn, is likely to require significant host control capabilities in order to ensure effective workload processing in the face of dynamically varying job and data assignments.

At the present time, work on centralized host network control capabilities is only beginning. 24 Their absence reflects the need for control strategies concerned with basic issues of: scheduling, major software capabilities and the basic hardware architecture of a system. In a single site computing environment, control strategies tended to be a byproduct of control tactics used for fine tuning the system. 29 Such tactics were primarily implemented by systems programmers having detailed knowledge of the system, its workload, anticipated needs, and the informal pecking order of customer priorities and were oriented toward a fine tuning of the system in a manner appropriate to such knowledge. It follows that tactics are relatively unsuited to centralized implementation while, as discussed in Reference 24, the opportunities for a centralized implementation of control strategies seem reasonable. A clear requirement for an effective control strategy is a careful, guaranteed delivery of system resources. An innovative approach to this subject is described in Reference 26.

Network (host plus subnetwork) control is clearly destined to be a subject of extreme importance in the context of Mission Oriented Networks. Existence of network control capabilities permits development and control of geographically dispersed data bases, tradeoffs between computing and communication, and assurance that remote users receive service comparable to that experienced by local users.

**DATA MIGRATION**

Data migration, as discussed in the Introduction, provides the basic capability required to permit a process executing in one computer to access data contained in another computer. Current data migration capabilities require that such access be accomplished via
transmission of the file to the site at which the data is
to be processed. Although this approach is, perhaps,
reasonable in the context of an individual computer
system, it seems unlikely to be acceptable in a data
processing environment in which, typically, the owner
of a file is unwilling to permit users to copy the file. In
addition, for reasons of security and privacy, the
owner may also be unwilling to let an accessing user
access the entire file and may instead wish to restrict
access to limited portions of the file.

Development of a general data migration capability
requires three basic capabilities:

- data selection,
- data translation, and
- data transformation.

Data selection provides the basic capability (daemon)
for remote accessing of data at the subfile level. As
such, it should be envisioned as a transaction processor
colocated with the data which, upon issuance of an
appropriate request by a remote process, accesses a
description of the file (specified in some appropriate
data description language), and reads the requested
record(s) in order to transmit them to the requesting
process. At a conceptual level, it follows that the ac­
cessing process could therefore treat remote data just
as it treats local data. At a practical level this observ­
a tion needs to be tempered by the knowledge that net­
work accesses will, in general, consume significantly
more time than local data accesses and, as a result, ac­
cess delays must be factored in to determine the rela­
tive desirability of utilizing a data selection capability
versus transmission of the entire file. From an imple­
mentation viewpoint, it is of interest to observe that
the basic data selection capabilities required relate
closely to those required to support data translation. Moreover, these capabilities also seem required to im­
plement an effective locking capability in a networking
environment.

It is well recognized that hosts in a heterogeneous
network use different bit patterns for encoding infor­
mation. Data translation is the basic capability which
permits hosts to communicate with each other in spite
of these differences. It follows that a data translation
capability is central to any effective capability to com­
 municate among heterogeneous computers.

Data, particularly in a data processing environment,
is usually stored as a collection of structured records.
Effective transmission of data therefore requires aug­
mentation of data translation to encompass preserva­
tion of record structure. The prerequisite capability for
such preservation is an ability to describe the record
structure and this, in turn, requires Data Description
Languages.

Data description languages can be structured into
Logical Data Description Languages (LDDLs) con­
cerned with describing the logical structure of the data
and with characterizing fields, and Physical Data De­
scription Languages (PDDLs) which serve as the
mechanism for describing how the data is physically
laid out on the storage media.

LDDLs are relatively straightforward and can be
pursued at various levels of abstraction as discussed in
References 30, 31 and 32. PDDLs are also logically
straightforward but, from a user viewpoint, tedious.
However, in a non-networking environment, PDDLs
are clearly required to permit one system to read data
written by another system. In contrast, in a net­
working environment, it seems feasible to eliminate the need
for PDDLs and rely upon the system access routines to
access the requested data and via a utility program,
translate it into an appropriate "normal" form for
transmission to the requesting site where it is again
retranslated from the "normal" form into an appro­
priate local form. An exploratory discussion of this
approach is contained in the paper: An Approach To
Data Migration in Computer Networks. It follows that
data translation in a networking environment seems
substantially simpler than translation in a non-net­
working environment.

Data transformation provides the basic capability for
restructuring the logical form of data into a form more
appropriate to the needs of the user or, alternatively,
one which the owner of the data is willing to have
transmitted. (For example, the owner may be unwill­
ing to permit certain fields such as salary information
to be transmitted.)

Data transformation requires, in general, three basic
capabilities:

- arithmetic operations,
- logical operations,
- string operations.

Moreover, to be useful in a generalized data base en­
vironment, data transformation should permit opera­tion on multiple input streams to produce multiple output
streams.

Currently, some work is being done on the general
topic of data transformation. Moreover, a general
single input stream single output stream data transfor­
mation capability was implemented within ARPANET
in the context of the Data Reconfiguration Service.
This system, in view of the primitive nature of the user
interface and the highly procedural nature of the trans­
formations could more properly be regarded as an ex­
periment demonstrating the viability of the concept
and, in this perspective should be viewed as a success.
However, to be useful as a general purpose tool, sub­
stantially more sophisticated interface capabilities are
required. Moreover, it would be generally desirable to
permit both distributed and centralized implementa­
tions of this capability.

In summary, the basic needs required to support a
data migration capability are reasonably clear and can
be structured in a fairly coherent manner. Moreover,
many of the required capabilities exist in various rud­
imentary forms at the present time.
NETWORK JOB EXECUTION

We define a network job to consist of a lattice (tied tree) of job steps which, as illustrated in Figure 1, may each be executed on a distinct and possibly dissimilar host. The characteristics of a network job are determined by those of the individual job steps.

In the simplest case, a job step may be a "batch" step which, once initiated, requires no further interaction with the NOS until termination. Support of the execution requirements of such a step is relatively straightforward.

In general, job steps executing in a networking environment may be anticipated to have more sophisticated support requirements including:

- user interaction,
- remote procedure calls, and
- synchronization with remote processes.

(It should be noted that execution of a network job is often viewed as an interprocess communication issue and thus a process rather than a job viewpoint is adopted. This viewpoint is correct for an implementor; however, from the viewpoint of the user the issue is really whether a remote job can be executed as is evident from the discussion of the Call-Return mechanism in Reference 11 as the basic primitive provided to a user.)

Support of user interaction with a network job step may prove relatively difficult in some operating systems since it tends to conflict with the desire for encapsulation of the job step which is useful in carefully controlling the interaction of the job step with the local host support capabilities such as the file system.

Provision of a capability to support remote procedure calls in a manner analogous to that in which local procedure calls are supported is clearly desirable. One approach to the development of such a capability is described in the paper A High-Level Framework For Network-Based Resource Sharing. In this paper it is observed that this capability is particularly desirable to provide an alternative means to the more general issue of protocol implementation within a computer network.

Support of remote procedure calls carries, by analogy with the characteristics of local procedure calls, the implication that the calling process will terminate execution pending satisfaction of the call. In general, this need not be the case and, in view of the relatively large delays likely to occur in calls to remote systems (on the order of a few hundred milliseconds minimum due to subnetwork delays), more sophisticated capabilities are desirable to permit concurrent execution of local and remote processes. Thus, synchronization capabilities are required.

Network job execution requirements

Execution of a network job requires four major capabilities:

- job step assignment and control,
- step execution monitoring,
- JCL Generation, and
- Interprocess Communication Support.

Job step assignment and control is required for assigning job steps to processors: arranging for job step initiation upon satisfaction of all appropriate precedence and priority constraints; migration of files as required to permit job step initiation; and communication with the user. Assignment of job steps clearly requires a capability to detect any existing (step) parallelism. Moreover, such assignment may be made at the direction of the user, at the request of the host (to reduce peak workload), or at the suggestion of some centralized network management capability to balance global resource requests against resource availability.

Step execution monitoring is required to interface the scheduling of network originated jobs with local jobs. In particular, it provides step initiation; monitors step execution to provide restart and, potentially, recovery capabilities; and notifies the job step assignment routines of the termination of one step to permit initiation of successor steps. To support the needs of a job step to communicate with remote data, the appropriate data paths with data selectors stored in remote hosts must also be arranged.

Network JCL

A job control language provides four generic capabilities:

- identifying the precedence and priority conditions required for job step execution,
- making files within a user's directory known to
the step which usually has its own expectations regarding the naming of the files,

• inserting files generated by a program into the appropriate directory, and

• controlling the assignment of files to devices and, for a given file, controlling the layout of the file on the device.

Powerful job control languages, such as are commonly available with the larger data processing systems provide great power in effecting these functions and, as a result, are noted for the complexity of the statements required to achieve relatively simple (compile-load-go) objectives. In contrast, the JCL's provided with many of the scientifically oriented computer systems are quite simple and reflect a more limited spectrum of potential user needs. An intermediate point in this spectrum is provided by the capability which some of the more sophisticated JCL's provide of using "canned" procedures to achieve relatively straightforward objectives. In this context, it seems likely that interactive approaches to JCL generation can be established which expedite generation of the appropriate JCL for the broad majority of user requirements. Users interested in maximum flexibility will probably still have to fend for themselves with the appropriate JCL. That is, user utilization of remote systems can be rendered no simpler than local usage if access to the full spectrum of system capabilities is to be provided.

Network IPC

Interprocess communication is required both as a technical prerequisite to building a network operating system and as a capability to be provided to users of the system to enable them to take full advantage of the opportunities (remote procedure calls, parallel execution, synchronized execution) afforded by the network. User IPC primitives are required for data transmission and process synchronization and control. Data primitives were discussed earlier. Synchronization and control primitives required to call procedures or processes, transfer data, and synchronize resource utilization can be divided into four major types:

• signal/wait-signal,

• wait/no-wait,

• transfer-control/retain-control, and

• preserve-state/reset-state.

The first primitive is required to permit a process to signal another process or procedure advertising its intentions; wait-signal is then required in order to permit a process to enter the wait state until receipt of an appropriate signal. The second primitive determines if parallel execution of both caller and called processes (entities) is to be supported The third primitive supports coroutines—in which both the calling and the called process have equal status—and the fourth capability is also required to support coroutine calls since the return location depends on the call location.

IPC capabilities can be provided in varying levels of sophistication reflecting: quality of the user interface, error recovery capabilities, and permissible complexity in the parameter string passed to the called procedure or process. Careful investigations of the total environment desirable to support effective and efficient IPC have been undertaken. Effectiveness issues are discussed in Reference 11, while issues of efficiency are discussed in Reference 37. Moreover, alternative message based implementation strategies to the connection oriented ARPANET approach to inter-process communication are described in Reference 38.

EXISTING RELATED CAPABILITIES

An overview of Network Operating Systems would not be complete without a discussion of currently existing capabilities. Currently, three such major capabilities can be distinguished:

• ARPANET Protocols,

• RSEXEC, and

• National Software Works.

ARPANET Protocols

ARPANET protocols, as illustrated in Figure 2, can be structured in a hierarchical manner. The first level of user oriented protocols encompasses the three major function oriented protocols: TELNET, File Transfer Protocol (FTP), and Data Reconfiguration Service (DRS). The capabilities provided by these protocols have been discussed in the preceding sections and the salient factor from the viewpoint of an NOS is the observation that the protocol only standardizes the communication path between a user process (with which

From the collection of the Computer History Museum (www.computerhistory.org)
the user interacts) and a server process (which effects the necessary actions). Thus, these protocols at a minimum fail to provide program callability and require some degree of reprogramming for its achievement.

To provide more directly applicable user capabilities, ARPA Network protocols also contain several applications oriented protocols such as Graphics and the Network Voice Protocol which are oriented toward provision of a specific collection of functional capabilities. As an alternative approach, one may wish to consider implementation of a more general user programming environment such as is proposed in the paper: A High-Level Framework for Network-Based Resource Sharing.

RSEXEC

The Resource Sharing Executive (RSEXEC) provides a network file system for the TENEX subcollection of hosts within the ARPANET. As such, it provides Network Wide Directories and, in addition, has supported system modifications which enable automatic file migration for dynamically generated file calls to non-local files. The standard set of inter-file manipulation commands are also provided. However, in view of the homogeneous nature of the hosts, issues of data translation did not require consideration and the issues of data selection, data transformation, and network job execution (excluding network IPC) were not specifically attacked.

National software works

The National Software Works (NSW) is concerned with providing a program production capability distributed across a collection of hosts. It thereby potentially permits utilization of more sophisticated, relatively non-portable tools than would be possible if all tools were forced to execute at a single host.

Achievement of the NSW objectives requires careful consideration of many of the issues which are relevant in the design of a Network Operating System. However, in view of its more limited objective, substantially more sophisticated capabilities can be provided. In particular, in supporting tool access within the NSW framework, it proves feasible to render the host operating system apparently invisible to the user through effective encapsulation of the user and all interactions with the tool. Thus, the mediation role provided by an NOS is augmented to encompass encapsulation. In view of the magnitude of the NSW project we defer a more detailed discussion of NSW capabilities and refer the reader to Reference 5.

CONCLUDING REMARKS

In summary, we feel we have established the major objectives and goals underlying development of a general purpose Network Operating System. Although space limitations preclude a general discussion of their feasibility and manner of attainment, the interested reader may consult a study directed to this end. As discussed therein, we feel it is reasonable to conclude that the objectives established are technically feasible; their utility is clearly manifest; and substantial portions of the required technological capabilities currently exist.

REFERENCES


20. —— A Tutorial for the Electronic Notebook Conference (TEN-C) System on ARPANET, ISI/R-75-38, USC/Information Sciences Institute, Marina del Rey, California, August 1975.


A high-level framework for network-based resource sharing*

by JAMES E. WHITE
Stanford Research Institute
Menlo Park, California

ABSTRACT
This paper proposes a high-level, application-independent framework for the construction of distributed systems within a resource sharing computer network. The framework generalizes design techniques in use within the ARPA Computer Network. It eliminates the need for application-specific communication protocols and support software, thus easing the task of the applications programmer and so encouraging the sharing of resources. The framework consists of a network-wide protocol for invoking arbitrary named functions in a remote process, and machine-dependent system software that interfaces one applications program to another via the protocol. The protocol provides mechanisms for supplying arguments to remote functions and for retrieving their results; it also defines a small number of standard data types from which all arguments and results must be modeled. The paper further proposes that remote functions be thought of as remotely callable subroutines or procedures. This model would enable the framework to more gracefully extend the local programming environment to embrace modules on other machines.

THE GOAL, RESOURCE SHARING
The principal goal of all resource-sharing computer networks, including the now international ARPA Network (the ARPANET), is to usefully interconnect geographically distributed hardware, software, and human resources. Achieving this goal requires the design and implementation of various levels of support software within each constituent computer, and the specification of network-wide "protocols" (that is, conventions regarding the format and the relative timing of network messages) governing their interaction. This paper outlines an alternative to the approach that ARPANET system builders have been taking since work in this area began in 1970, and suggests a strategy for modeling distributed systems within any large computer network.

The first section of this paper describes the prevailing ARPANET protocol strategy, which involves specifying a family of application-dependent protocols with a network-wide inter-process communication facility as their common foundation. In the second section, the application-independent command/response discipline that characterizes this protocol family is identified and its isolation as a separate protocol proposed. Such isolation would reduce the work of the applications programmer by allowing the software that implements the protocol to be factored out of each applications program and supplied as a single, installation-maintained module. The final section of this paper proposes an extensible model for this class of network interaction that in itself would even further encourage the use of network resources.

THE CURRENT SOFTWARE APPROACH TO RESOURCE SHARING

Function-oriented protocols

The current ARPANET software approach to facilitating resource sharing has been detailed elsewhere in the literature. Briefly, it involves defining a Host-Host Protocol by which the operating systems of the various "host" computers cooperate to support a network-wide inter-process communication (IPC) facility, and then various function-oriented protocols by which processes deliver and receive specific services via IPC. Each function-oriented protocol regulates the dialog between a resident "server process" providing the service, and a "user process" seeking the service on behalf of a user (the terms "user" and "user process" will be used consistently throughout this paper to distinguish the human user from the computer process acting on his behalf).

The current Host-Host Protocol has been in service since 1970. Since its initial design and implementation, a variety of deficiencies have been recognized and several alternative protocols suggested. Although improvements at this level would surely have a positive effect upon Network resource sharing, the present paper simply assumes the existence of some form of

* The work reported here was supported by the Advanced Research Projects Agency of the Department of Defense, and by the Rome Air Development Center of the Air Force.
IPC and focuses attention upon higher level protocol design issues.

Each of the function-oriented protocols mentioned in this paper constitutes the official ARPANET protocol for its respective application domain and is therefore implemented at nearly all of the 75 host installations that now comprise the Network. It is primarily upon this widely implemented protocol family (and the philosophy it represents) that the present paper focuses. Needless to say, other important resource-sharing tools have also been constructed within the ARPANET. The Resource-Sharing Executive (RSEXEC), designed and implemented by Bolt, Beranek and Newman, Inc., provides an excellent example of such work.

Experience with and limitations of hands-on resource sharing

The oldest and still by far the most heavily used function-oriented protocol is the Telecommunications Network protocol (TELNET), which effectively attaches a terminal on one computer to an interactive time-sharing system on another, and allows a user to interact with the remote system via the terminal as if he were one of its local users.

As depicted in Figure 1, TELNET specifies the means by which a user process monitoring the user’s terminal is interconnected, via an IPC communication channel, with a server process with access to the target time-sharing system. TELNET also legislates a standard character set in which the user’s commands and the system’s responses are to be represented in transmission between machines. The syntax and semantics of these interchanges, however, vary from one system to another and are unregulated by the protocol; the user and server processes simply shuttle characters between the human user and the target system.

Although the hands-on use of remote resources that TELNET makes possible is a natural and highly visible form of resource sharing, several limitations severely reduce its long-term utility:

1. It forces upon the user all of the trappings of the resource’s own system.

To exploit a remote resource, the user must leave the familiar working environment provided by his local system and enter an alien one with its own peculiar system structure (login, logout, and subsystem entry and exit procedures) and command language discipline (command recognition and completion conventions, editing characters, and so on). Hands-on resource sharing thus fails to provide the user with the kind of organized and consistent workshop he requires to work effectively.

2. It provides no basis for bootstrapping new composite resources from existing ones.

Because the network access discipline imposed by each resource is a human-engineered command language, rather than a machine-oriented communication protocol, it is virtually impossible for one resource to programatically draw upon the services of others. Doing so would require that the program deal successfully with complicated echoing and feedback characteristics; unstructured, even unsolicited system responses; and so forth. Hands-on resource sharing thus does nothing to provide an environment in which existing resources can be used as building blocks to construct new, more powerful ones.

These inherent limitations of hands-on resource sharing are removed by a protocol that simplifies and standardizes the dialog between user and server processes. Given such a protocol, the various remote resources upon which a user might wish to draw can indeed be made to appear as a single, coherent workshop by interposing between him and them a command language interpreter that transforms his commands into the appropriate protocol utterances. The construction of composite resources also becomes feasible, since each resource is accessible by means of a machine-oriented protocol and can thus be readily employed by other processes within the network.

Standardizing the inter-machine dialog in specific application areas

After the TELNET protocol had been designed and widely implemented within the ARPANET, work began on a family of function-oriented protocols designed for use by programs, rather than human users. Each such protocol standardizes the inter-machine dialog in a particular application area. While TELNET dictates...
only the manner in which user and server processes are
interconnected via the IPC facility, and the character
set in which the two processes communicate once con­
nected, each member of this family specifies in addition
the syntax and semantics of the commands and re­
sponses that comprise their dialog.

Protocols within this family necessarily differ in
substance, each specifying its own application-specific
command set. The File Transfer Protocol (FTP), 12
for example, specifies commands for manipulating files,
and the Remote Job Entry Protocol (RJE) 13 specifies
commands for manipulating batch jobs. Protocols
throughout the family are, however, similar in form,
each successive family member having simply inherited
the physical features of its predecessors. Thus FTP
and RJE enforce the same conventions for formulating
commands and responses.

This common command/response discipline requires
that commands and responses have the following re­
spective formats:

command-name <SP> parameter <CRLF>
response-number <SP> text <CRLF>

Each command invoked by the user process is identi­
fied by NAME and is allowed a single PARAMETER. Each response generated by the server process contains
a three-digit decimal response NUMBER (to be inter­
preted by the user process) and explanatory TEXT
(for presentation, if necessary, to the user). Response
numbers are assigned in such a way that, for example,
positive and negative acknowledgments can be easily
distinguished by the user process.

FTP contains, among others, the following com­
mands (each listed with one of its possible responses)
for retrieving, appending to, replacing, and deleting
files, respectively, within the server process file
system:

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETR &lt;SP&gt; filename</td>
<td>250 &lt;SP&gt; Beginning transfer. &lt;CRLF&gt;</td>
</tr>
<tr>
<td>APPE &lt;SP&gt; filename</td>
<td>400 &lt;SP&gt; Not implement. &lt;CRLF&gt;</td>
</tr>
<tr>
<td>STOR &lt;SP&gt; filename</td>
<td>453 &lt;SP&gt; Directory/flow. &lt;CRLF&gt;</td>
</tr>
<tr>
<td>DELE &lt;SP&gt; filename</td>
<td>450 &lt;SP&gt; File not found. &lt;CRLF&gt;</td>
</tr>
</tbody>
</table>

A COMMAND/RESPONSE PROTOCOL, THE BASIS
FOR AN ALTERNATIVE APPROACH

The importance of factoring out the command/-
response discipline

That FTP, RJE, and the other protocols within this
family share a common command/response discipline
is a fact not formally recognized within the protocol
literature, and each new protocol document describes
it in detail, as if for the first time. Nowhere are these
conventions codified in isolation from the various con­
texts in which they find use, being viewed as a neces­
sary but relatively unimportant facet of each function­
oriented protocol. "This common command/response
discipline has thus gone unrecognized as the important,
application-independent protocol that it is."

This oversight has had two important negative
effects upon the growth of resource sharing within the
ARPANET:

1) It has allowed the command/response discipline
to remain crude.

As already noted, operations that require
more than a single parameter are consistently
implemented as two or more separate com­
mands, each of which requires a response and
thus incurs the overhead of a full round-trip
network delay. Furthermore, there are no
standards for encoding parameter types other
than character strings, nor is there provision
for returning results in a command response.

2) It has placed upon the applications programmer
the burden of implementing the network “run­
time environment (RTE)" that enables him to
access remote processes at the desired, func­
tional level.

Before he can address remote processes in
terms like the following:

`execute function DELE with argument TEXT-
FILE on machine X`

the applications programmer must first con­
struct (as he invariably does in every program
he writes) a module that provides the desired
program interface while implementing the
agreed upon command/response discipline. This
run-time environment contains the code re­
quired to properly format outgoing commands,
to interface with the IPC facility, and to parse
incoming responses. Because the system pro­
vides only the IPC facility as a foundation, the
applications programmer is deterred from using remote resources by the amount of specialized knowledge and software that must first be acquired.

If, on the other hand, the command/response discipline were formalized as a separate protocol, its use in subsequent function-oriented protocols could rightly be anticipated by the systems programmer, and a single run-time environment constructed for use throughout an installation (in the worst case, one implementation per programming language per machine might be required). This module could then be placed in a library and, as depicted in Figure 2, link loaded with (or otherwise made available to) each new applications program, thereby greatly simplifying its use of remote resources.

Furthermore, since enhancements to it would pay dividends to every applications program employing its services, the run-time environment would gradually be augmented to provide additional new services to the programmer.

The thesis of the present paper is that one of the keys to facilitating network resource sharing lies in (1) isolating as a separate protocol the command/response discipline common to a large class of applications protocols; (2) making this new, application-independent protocol flexible and efficient; and (3) constructing at each installation a RTE that employs it to give the applications programmer easy and high-level access to remote resources.

**Specifications for the command/response protocol**

Having argued the value of a command/response protocol (hereafter termed the Protocol) as the foundation for a large class of applications protocols, there remains the task of suggesting the form that the Protocol might take. There are eight requirements. First, it must reproduce the capabilities of the discipline it replaces:

1. Permit invocation of arbitrary, named commands (or functions) implemented by the remote process.
2. Permit command outcomes to be reported in a way that aids both the program invoking the command and the user under whose control it may be executing.

Second, the Protocol should remove the known deficiencies of its predecessor, that is:

3. Allow an arbitrary number of parameters to be supplied as arguments to a single command.
4. Provide representations for a variety of parameter types, including but not limited to character strings.
5. Permit commands to return parameters as results as well as accept them as arguments.

And, finally, the Protocol should provide whatever additional capabilities are required by the more complex distributed systems whose creation the Protocol seeks to encourage. Although others may later be identified, the three capabilities below are recognized now to be important:

6. Permit the server process to invoke commands in the user process, that is, eliminate entirely the often inappropriate user/server distinction, and allow each process to invoke commands in the other.

   In the workshop environment alluded to earlier, for example, the user process is the command language interpreter and the server process is any of the software tools available to the user. While most commands are issued by the interpreter and addressed to the tool, occasionally the tool must invoke commands in the interpreter or in another tool. A graphical text editor, for example, must invoke commands within the interpreter to update the user's display screen after an editing operation.

7. Permit a process to accept two or more commands for concurrent execution.

   The text editor may wish to permit the user to initiate a long formatting operation with one command and yet continue to issue additional, shorter commands before there is a response to the first.

8. Allow the process issuing a command to suppress the response the command would otherwise elicit.

   This feature would permit network traffic to be reduced in those cases in which the process invoking the command deems a response unnecessary. Commands that always succeed but never return results are obvious candidates for this kind of treatment.
A formulation of the protocol that meets these specifications

The eight requirements listed above are met by a protocol in which the following two messages are defined:

- message-type = COMMAND [tid] command-name arguments
- message-type = RESPONSE tid outcome results

Here and in subsequent protocol descriptions, elements enclosed in square brackets are optional.

The first message invokes the command whose NAME is specified using the ARGUMENTS provided. The second is issued in eventual response to the first and returns the OUTCOME and RESULTS of the completed command. Whenever OUTCOME indicates that a command has failed, the command's RESULTS are required to be an error number and diagnostic message, the former to help the invoking program determine what to do next, the latter for possible presentation to the user. The protocol thus provides a framework for reporting errors, while leaving to the applications program the tasks of assigning error numbers and composing the text of error messages.

There are several elements of the Protocol that are absent from the existing command/response discipline:

1. RESULTS, in fulfillment of Requirement 5.
2. A MESSAGE TYPE that distinguishes commands from responses, arising from Requirement 6.

   In the existing discipline, this distinction is implicit, since user and server processes receive only responses and commands, respectively.
3. An optional transaction identifier TID by which a command and its response are associated, arising from Requirements 7 and 8.

   The presence of a transaction identifier in a command implies the necessity of a response echoing the identifier; and no two concurrently outstanding commands may bear the same identifier.

Requirements 3 and 4—the ability to transmit an arbitrary number of parameters of various types with each command or response—are most economically and effectively met by defining a small set of primitive "data types" (for example, booleans, integers, character strings) from which concrete parameters can be modeled, and a "transmission format" in which such parameters can be encoded. Appendix A suggests a set of data types suitable for a large class of applications; Appendix B defines some possible transmission formats.

The protocol description given above is, of course, purely symbolic. Appendix C explores one possible encoding of the Protocol in detail.

Summarizing the arguments advanced so far

The author trusts that little of what has been presented thus far will be considered controversial by the reader. The following principal arguments have been made:

1. The more effective forms of resource sharing depend upon remote resources being usefully accessible to other programs, not just to human users.
2. Application-dependent protocols providing such access using the current approach leave to the applications programmer the task of constructing the additional layer of software (above the IPC facility provided by the system) required to make remote resources accessible at the functional level, thus discouraging their use.
3. A single, resource-independent protocol providing flexible and efficient access at the functional level to arbitrary remote resources can be devised.
4. This protocol would make possible the construction at each installation of an application-independent, network run-time environment making remote resources accessible at the functional level and thus encouraging their use by the applications programmer.

A protocol as simple as that suggested here has great potential for stimulating the sharing of resources within a computer network. First, it would reduce the cost of adapting existing resources for network use by eliminating the need for the design, documentation, and implementation of specialized delivery protocols. Second, it would encourage the use of remote resources by eliminating the need for application-specific interface software. And finally, it would encourage the construction of new resources built expressly for remote access, because of the ease with which they could be offered and used within the network software marketplace.

A HIGH-LEVEL MODEL OF THE NETWORK ENVIRONMENT

The importance of the model imposed by the protocol

The Protocol proposed above imposes upon the applications programmer a particular model of the network environment. In a heterogeneous computer network, nearly every protocol intended for general implementation has this effect, since it idealizes a class of operations that have concrete but slightly different equivalents in each system. Thus the ARPANET's TELNET Protocol alluded to earlier, for example, specifies a Network Virtual Terminal that attempts to provide a best fit to the many real terminals in use around the Network.
As now formulated, the Protocol models a remote resource as an interactive program with a simple, rigidly specified command language. This model follows naturally from the fact that the function-oriented protocols from which the Protocol was extracted were necessitated by the complexity and diversity of user-oriented command languages. The Protocol may thus legitimately be viewed as a vehicle for providing, as an adjunct to the sophisticated command languages already available to users, a family of simple command languages that can readily be employed by programs.

While the command/response model is a natural one, others are possible. A remote resource might also be modeled as a process that services and replies to requests it receives from other computer processes. This request/reply model would emphasize the fact that the Protocol is a vehicle for inter-process communication and that no human user is directly involved.

Substituting the request/reply model for the command/response model requires only cosmetic changes to the Protocol:

message-type = REQUEST [tid] op-code
arguments
message-type = REPLY tid outcome results

In the formulation above, the terms “REQUEST”, “REPLY”, and “op-code” have simply been substituted for “COMMAND”, “RESPONSE”, and “command-name”, respectively.

The choice of model need affect neither the content of the Protocol nor the behavior of the processes whose dialog it governs. Use of the word “command” in the command/response model, for example, is not meant to imply that the remote process can be coerced into action. Whatever model is adopted, a process has complete freedom to reject an incoming remote request that it is incapable of or unwilling to fulfill.

But even though it has no substantive effect upon the Protocol, the selection of a model—command/response, request/reply, and so on—is an important task because it determines the way in which both applications and systems programmers perceive the network environment. If the network environment is made to appear foreign to him, the applications programmer may be discouraged from using it. The choice of model also constrains the kind and range of protocol extensions that are likely to occur to the systems programmer; one model may suggest a rich set of useful extensions, another lead nowhere (or worse still, in the wrong direction).

In this final section of the paper, the author suggests a network model (hereafter termed the Model) that he believes will both encourage the use of remote resources by the applications programmer and suggest to the systems programmer a wide variety of useful Protocol extensions. Unlike the substance of the Protocol, however, the Model has already proven quite controversial within the ARPANET community.

Modeling resources as collections of procedures

Ideally, the goal of both the Protocol and its accompanying RTE is to make remote resources as easy to use as local ones. Since local resources usually take the form of resident or library subroutines, the possibility of modeling remote commands as “procedures” immediately suggests itself. The Model is further confirmed by the similarity that exists between local procedures and the remote commands to which the Protocol provides access. Both carry out arbitrarily complex, named operations on behalf of the requesting program (the caller); are governed by arguments supplied by the caller; and return to it results that reflect the outcome of the operation. The procedure call model thus acknowledges that, in a network environment, programs must sometimes call subroutines in machines other than their own.

Like the request/reply model already described, the procedure call model requires only cosmetic changes to the Protocol:

message-type = CALL [tid] procedure-name
arguments
message-type = RETURN tid outcome results

In this third formulation, the terms “CALL”, “RETURN”, and “procedure-name” have been substituted for “COMMAND”, “RESPONSE”, and “command-name”, respectively. And in this form, the Protocol might aptly be designated a “procedure call protocol (PCP)”.

The procedure call model would elevate the task of creating applications protocols to that of defining procedures and their calling sequences. It would also provide the foundation for a true distributed programming system (DPS) that encourages and facilitates the work of the applications programmer by gracefully extending the local programming environment, via the RTE, to embrace modules on other machines.” This integration of local and network programming environments can even be carried as far as modifying compilers to provide minor variants of their normal procedure-calling constructs for addressing remote procedures (for which calls to the appropriate RTE primitives would be dropped out).

Finally, the Model is one that can be naturally extended in a variety of ways (for example, coroutine linkages and signals) to further enhance the distributed programming environment.

Clarifying the procedure call model

Although in many ways it accurately portrays the class of network interactions with which this paper...
deals, the Model suggested above may in other respects tend to mislead the applications programmer. The Model must therefore be clarified:

(1) Local procedure calls are cheap; remote procedure calls are not.

Local procedure calls are often effected by means of a single machine instruction and are therefore relatively inexpensive. Remote procedure calls, on the other hand, would be effected by means of a primitive provided by the local RTE and require an exchange of messages via IPC.

Because of this cost differential, the applications programmer must exercise discretion in his use of remote resources, even though the mechanics of their use will have been greatly simplified by the RTE. Like virtual memory, the procedure call model offers great convenience, and therefore power, in exchange for reasonable alertness to the possibilities of abuse.

(2) Conventional programs usually have a single locus of control; distributed programs need not.

Conventional programs are usually implemented as a single process with exactly one locus of control. A procedure call, therefore, traditionally implies a transfer of control from caller to callee. Distributed systems, on the other hand, are implemented as two or more processes, each of which is capable of independent execution. In this new environment, a remote procedure call need not suspend the caller, which is capable of continuing execution in parallel with the called procedure.

The RTE can therefore be expected to provide, for convenience, two modes of remote procedure invocation: a blocking mode that suspends the caller until the procedure returns; and a non-blocking mode that releases the caller as soon as the CALL message has been sent or queued. Most conventional operating systems already provide such a mode choice for I/O operations. For non-blocking calls, the RTE must also, of course, arrange to asynchronously notify the program when the call is complete, or provide an additional primitive by which the applications program can periodically test for that condition.

Finally, the applications programmer must recognize that by no means all useful forms of network communication are effectively modeled as procedure calls. The lower level IPC facility that remains directly accessible to him must therefore be employed in those applications for which the procedure call model is inappropriate and RTE-provided primitives simply will not do.

SOME EXPECTATIONS

Both the Procedure Call Protocol and its associated Run-Time Environment have great potential for facilitating the work of the network programmer; only a small percentage of that potential has been discussed in the present paper. Upon the foundation provided by PCP can be erected higher level application-independent protocol layers that further enhance the distributed programming environment by providing even more powerful capabilities (see Appendix D).

As the importance of the RTE becomes fully evident, additional tasks will gradually be assigned to it, including perhaps those of:

(1) Converting parameters between the format employed internally by the applications program, and that imposed by the Protocol.

(2) Automatically selecting the most appropriate inter-process transmission format on the basis of the two machines' word sizes.

(3) Automatically substituting for network IPC a more efficient form of communication when both processes reside on the same machine.

The RTE will eventually offer the programmer a wide variety of application-independent, network-programming conveniences, and so, by means of the Protocol, become an increasingly powerful distributed-system-building tool.

ACKNOWLEDGMENTS

Many individuals within both SRI's Augmentation Research Center (ARC) and the larger ARPANET community have contributed their time and ideas to the development of the Protocol and Model described in this paper. The contributions of the following individuals are expressly acknowledged: Dick Watson, Jon Postel, Charles Irby, Ken Victor, Dave Maynard, and Larry Garlick of ARC; and Bob Thomas and Rick Schantz of Bolt, Beranek and Newman, Inc.

ARC has been working toward a high-level framework for network-based distributed systems for a number of years now. The particular Protocol and Model described here result from research begun by ARC in July of 1974. This research included developing the Model; designing and documenting the Protocol required to support it; and designing, documenting, and implementing a prototype run-time environment for a particular machine, specifically a PDP-10 running the Tenex operating system developed by Bolt, Beranek and Newman, Inc.

Three design iterations were carried out during a 12-month period, and the resulting specification implemented for Tenex. The Tenex RTE provides a superset of the capabilities presented in the body of this paper and Appendices A through C as well as those alluded to in Appendix D.
REFERENCES


APPENDIX A—SUGGESTED DATA TYPES

The Protocol requires that every parameter or "data object" be represented by one of several primitive data types defined by the Model. The set of data types below is sufficient to conveniently model a large class of data objects, but since the need for additional data types (for example, floating-point numbers) will surely arise, the set must remain open-ended. Throughout the descriptions below, N is confined to the range [0, 2**15-1].

LIST: A list is an ordered sequence of N data objects called "elements". A LIST may contain other LISTS as elements, and can therefore be employed to construct arbitrarily complex composite data objects.

CHARSTR: A character string is an ordered sequence of N ASCII characters, and conveniently models a variety of textual entities, from short user names to whole paragraphs of text.

BITSTR: A bit string is an ordered sequence of N bits and, therefore, provides a means for representing arbitrary binary data (for example, the contents of a word of memory).

INTEGER: An integer is a fixed-point number in the range [-2**31, 2**31-1], and conveniently models various kinds of numerical data, including time intervals, distances, and so on.

INDEX: An index is an integer in the range [1, 2**15-1]. As its name and value range suggest, an INDEX can be used to address a particular bit or character within a string, or element within a list. INDEXes have other uses as well, including the modeling of handles or identifiers for open files, created processes, and the like. Also, because of their restricted range, INDEXes are more compact in transmission than INTEGERS (see Appendix B).

BOOLEAN: A boolean represents a single bit of information, and has either the value true or false.

EMPTY: An empty is a valueless place holder within a LIST or parameter list.

APPENDIX B—SUGGESTED TRANSMISSION FORMATS

Parameters must be encoded in a standard transmission format before they can be sent from one process to another via the Protocol. An effective strategy is to define several formats and select the most appropriate one at run-time, adding to the Protocol a mechanism for format negotiation. Format negotiation would be another responsibility of the RTE and could thus be made completely invisible to the applications program.

Suggested below are two transmission formats. The first is a 36-bit binary format for use between 96-bit
machines, the second an 8-bit binary, “universal” format for use between dissimilar machines. Data objects are fully typed in each format to enable the RTE to automatically decode and internalize incoming parameters should it be desired to provide this service to the applications program.

PCPB36, For Use Between 36-Bit Machines

Bits 0-13 Unused (zero)
Bits 14-17 Data type
   EMPTY = 1 INTEGER = 4 LIST = 7
   BOOLEAN = 2 BITSTR = 5
   INDEX = 3 CHARSTR = 6
Bits 18-20 Unused (zero)
Bits 21-35 Value or length N
   EMPTY unused (zero)
   BOOLEAN 14 zero-bits + 1-bit value (TRUE = 1/FALSE = 0)
   INDEX unsigned value
   INTEGER unused (zero)
   BITSTR unsigned bit count N
   CHARSTR unsigned character count N
   LIST unsigned element count N

Bits 36- Value
   EMPTY unused (nonexistent)
   BOOLEAN unused (nonexistent)
   INDEX unused (nonexistent)
   INTEGER two’s complement full-word value
   BITSTR bit string + zero padding to word boundary
   CHARSTR ASCII string + zero padding to word boundary
   LIST element data objects

PCPB8, For Use Between Dissimilar Machines

Byte 0 Data type
   EMPTY = 1 INTEGER = 4 LIST = 7
   BOOLEAN = 2 BITSTR = 5
   INDEX = 3 CHARSTR = 6

Bytes 1- Value
   EMPTY unused (nonexistent)
   BOOLEAN 7 zero-bits + 1-bit value (TRUE = 1/FALSE = 0)
   INDEX 2-byte unsigned value
   INTEGER 4-byte two’s complement value
   BITSTR 2-byte unsigned bit count N + bit string + zero padding to byte boundary
   CHARSTR 2-byte unsigned character count N + ASCII string
   LIST 2-byte element count N + element data objects

APPENDIX C—A DETAILED ENCODING OF THE PROCEDURE CALL PROTOCOL

Although the data types and transmission formats detailed in the previous appendixes serve primarily as vehicles for representing the arguments and results of remote procedures, they can just as readily and effectively be employed to represent the commands and responses by which those parameters are transmitted. Taking this approach, one might model each of the two Protocol messages as a PCP data object, specifically a LIST whose first element is an INDEX message type. The following concise statement of the Protocol then results:

LIST (CALL, tid, procedure, arguments)
   INDEX = 1 INDEX/EMPTY CHARSTR LIST
LIST (RETURN, tid, outcome, results)
   INDEX = 2 INDEX BOOLEAN LIST

The RESULTS of an unsuccessful procedure would be represented as follows:

LIST (error, diagnostic)
   INDEX CHARSTR

APPENDIX D—A LOOK AT SOME POSSIBLE EXTENSIONS TO THE MODEL

The result of the distributed-system-building strategy proposed in the body of this paper and the preceding appendixes is depicted in Figure 3. At the core of each process is the inter-process communication facility provided by the operating system, which effects the transmission of arbitrary binary data between distant processes. Surrounding this core are conventions regarding first the format in which a few, primitive types of data objects are encoded in binary for IPC, and then the formats of several composite data objects (that is, messages) whose transmission either invokes or acknowledges the previous invocation of a remote procedure. Immediately above lies an open-ended protocol layer in which an arbitrary number of enhancements to the distributed programming environment can be implemented. Encapsulating these various protocol layers is the installation-provided runtime environment, which delivers DPS services to the applications program according to machine- and possibly programming-language-dependent conventions.

The Protocol proposed in the present paper recognizes only the most fundamental aspects of remote procedure calling. It permits the caller to identify the procedure to be called, supply the necessary arguments, determine the outcome of the procedure, and recover its results. In a second paper, the author proposes some extensions to this simple procedure call model, and attempts to identify other common forms of inter-process interaction whose standardization would
enhance the distributed programming environment. Included among the topics discussed are:

1. Coroutine linkages and other forms of communication between the caller and callee.
2. Propagation of notices and requests up the thread of control that results from nested procedure calls.
3. Standard mechanisms for remotely reading or writing system-global data objects within another program.
5. A standard means for creating and initializing processes, that is, for establishing contact with and logging into a remote machine, identifying the program to be executed, and so forth. This facility would permit arbitrarily complex process hierarchies to be created.
6. A mechanism for introducing processes to one another, that is, for superimposing more general communication paths upon the process hierarchy.

These and other extensions can all find a place in the open-ended protocol layer of Figure 3. The particular extensions explored in Reference 19 are offered not as dogma but rather as a means of suggesting the possibilities and stimulating further research.
Factors in interprocess communication protocol efficiency for computer networks

by CARL A. SUNSHINE
The Rand Corporation
Santa Monica, California

ABSTRACT

This paper considers the efficiency of interprocess communication protocols for distributed processing environments such as computer networks. Previous research has emphasized system performance at lower levels, within the communication medium itself, while this work examines requirements and performance of protocols for communication between processes in the Host computers attached to the communication system. Efficiency primarily concerns throughput and delay achievable for communication between remote processes. Various aspects of protocol operation are analyzed, and protocol policies concerning retransmission, flow control, buffering, acknowledgment, and packet size emerge as the most important factors in determining efficiency. Several graphs showing quantitative performance results for representative situations are included.

INTRODUCTION

The tremendous growth of computer communications in recent years has provided new problems as well as new opportunities in distributed processing. In particular, computer networks such as the ARPANET demand new techniques for providing reliable and efficient communication between processes running in different Host computers connected to the network. Typical network transmission characteristics include variable delay, limited bandwidth, and occasional loss, damage, duplication, or out-of-order delivery of messages. These transmission characteristics demand specially robust protocols, or algorithms and message formats for data exchange between remote processes, compared to mechanisms appropriate in a centralized system with common memory.

Protocols for such environments are usually based on transmission of packets containing data, control information such as sequence numbers, and a checksum for error detection. Correctly received packets are positively acknowledged by the destination. If no acknowledgment is received at the source within a given time-out period, packets are retransmitted. The reliability of such Positive Acknowledgment, Retransmission (PAR) protocols in the face of network transmission characteristics mentioned above has received increasing attention. This paper discusses major factors determining the efficiency of protocols for communication between remote processes over a packet switching network (PSN). Research on interprocess communication level protocols has just begun while related work on performance analysis of protocols used within a PSN is more abundant.

The two performance measures of primary interest are mean delay and mean throughput attainable. Throughput may be defined as the transmission rate of useful data between processes, excluding any control information or retransmissions required by the protocol. Delay is the time from starting to transmit a packet at the sender to successful arrival of the entire packet at the receiver in the case of one-way delay, or until arrival of an acknowledgment at the sender in the case of roundtrip delay. (Note that any waiting time between packet creation and start of transmission is not included in this delay definition.) Other efficiency performance measures of interest include retransmission rate (the proportion of packets which are retransmissions), line efficiency (the ratio of useful traffic to total traffic), and buffer requirements.

Numerous factors such as buffer allocation, receiver processing rate, flow control, error rates, error recovery techniques, header overhead, and network transmission delay and bandwidth help determine protocol performance. Some of these factors depend directly on communication network design and operation and represent essentially uncontrollable characteristics from the point of view of an interprocess communication protocol. Other factors depend on the behavior of processes communicating via the protocol.
This paper focuses on a third class of factors which are subject to control by the protocol itself. To provide efficient communication within the constraints of given network characteristics and user process behavior, a protocol can attempt to optimize several internal parameters such as retransmission interval, flow control strategy, buffering, acknowledgment scheme, and packet size. In the following sections, the impact of each of these parameters on protocol efficiency is explored. For this purpose, some simple mathematical models based on probability and queuing theory prove helpful. This paper discusses some of the results derived from these models, while Reference 5 presents a more complete treatment of the subject including full derivations of the results presented here.

RETRANSMISSION

As noted above, the primary purpose of retransmission is to overcome loss or damage of data packets (or acknowledgments) by the communication network. The protocol parameter controlling retransmission is the retransmission interval, \( R \). If no acknowledgment for a transmitted packet is received within time \( R \), the packet will be retransmitted.

The choice of \( R \) can have significant effects on both throughput and delay. A small \( R \) minimizes mean delay since lost packets are retransmitted promptly. If there is significant variation in network transmission delay for a packet, then quick retransmission may reduce mean delay even when no packets are lost or damaged. Larger \( R \), on the other hand, tends to maximize throughput since no bandwidth is "wasted" on unnecessary retransmissions. Assuming packets need not be delivered in order, there is no penalty for waiting until loss of a packet is certain before retransmitting it, since other packets can continue to be delivered.

To quantify these general observations, the following simple model proves useful. Let the network transmission delay be represented by a probability density function \( f(t) \) to allow for variation in delay. Both propagation time through the network (\( T_{\text{prop}} \)) and transmission time into the network (for a packet of length \( P \) and Host-to-network bandwidth of \( B \), this is \( P/B \)) are included in \( f(t) \). \( f(t) \) typically has a high peak around the "normal" delay, and a small but long tail representing the possibility of occasional long delays (see Figure 1). If \( f(t) \) represents roundtrip delay (time from transmitting a packet until an acknowledgment is received), then \( G(t) \) provides sufficient information to calculate the mean delay \( D_L \) until the first successful acknowledged transmission, and the mean number of transmissions \( N \). Since each successful transmission requires on the average \( N \) actual transmissions, throughput attainable will be proportional to a throughput factor \( T_{\text{Pretrans}} \) equal to the inverse of \( N \).

To demonstrate this analysis, let \( f(t) \) be the Erlangian family of distributions:

\[
f(t) = \frac{(k \cdot u \cdot t)^{k-1}}{(k-1)!} e^{-(k \cdot u \cdot t)}
\]

With an appropriate choice of the shape parameter \( k \) (e.g., \( k = 16 \)), this distribution provides an adequate model of typical network transmission delays\(^{12}\) (see Figure 2). Applying the above analysis to the Erlang-...
ian \( f(t) \) with \( k=16 \) gives Figure 3, a plot of throughput versus delay for varying retransmission interval \( R \) and several packet loss probabilities \( LS \) (the results for exponential \( f(t) \) are shown dotted for comparison). The definite “knee” in the curves for nonzero \( LS \) indicate an optimal value of \( R \). For larger \( R \), delay is increased with little savings in throughput. For smaller, \( R \), throughput is reduced (due to excessive retransmission) at little improvement in delay. This optimal value of \( R \) occurs at the time in the \( f(t) \) curve by which “most” transmissions would have succeeded (if no packets were lost or damaged). The optimal \( R \) is more clearly defined for \( f(t) \) with sharp peaks (more constant delay) and for higher packet loss probabilities.

To extend this analysis to protocols which perform sequencing, the assumption of independent identically distributed roundtrip delays for successive transmissions must be modified. In particular, loss of a packet or its acknowledgment causes acknowledgment of all subsequent packets to be delayed until the earlier error is corrected, even though other packets are successfully received. In this case, negative acknowledgments may improve performance by forcing prompt retransmission of the damaged packet, and suppressing retransmission of other outstanding packets (c.f. next section). Although negative acknowledgments help reduce performance losses, higher delay and lower throughput must be expected with a sequencing protocol if packets arrive significantly out of order.

FLOW CONTROL

Roundtrip delay is typically an order of magnitude greater than Host-to-network packet transmission time over a packet switching network. In this environment, a simple protocol that waits for acknowledgment of each packet before transmitting the next packet will be idle a large fraction of the time. Therefore, to achieve higher throughput, the sender must be allowed to transmit multiple packets before receiving any acknowledgments. Since each outstanding packet requires buffering and other communication resources, the amount of advance transmission required to achieve maximum throughput becomes an important efficiency question.

On the other hand, it is also necessary to limit the flow of information from source process to destination process. The communication network itself must guard against internal congestion by imposing “congestion control” constraints on traffic sources. More relevant to this study, each sender’s transmission rate should be matched to the receiver’s acceptance rate to minimize protocol resources required to support the given traffic. Achieving this matching is the primary purpose of a protocol’s flow control mechanisms.

Most flow control strategies can be described in terms of a limited window size of \( N \) packets (and/or bits) that may be transmitted but not yet acknowledged. When this limit is reached, the sender must stop transmitting new packets (although retransmissions of pending packets may proceed) until more transmission “credits” are returned by the receiver (often associated with acknowledgments).

This situation can be modeled as a dual-server closed queuing system with the transmitter as one server, and the network and receiver as the other. (This second server is an “infinite” server since packets can proceed in parallel through the network.) Service time at the transmitter (\( T_{local} \)) is the Host-to-network transmission time for a packet (\( P/B \)), while service time 2 (\( T_{net} \)) which includes propagation of the packet through the network, receiver processing, and return of an acknowledgment, is the roundtrip delay less service time 1. Define \( RHO \) as the ratio of service time 1 to service time 2. The window size \( N \) defines the number of customers (packets) in the system (see Figure 4).

The utilization of server 1 (\( UT \)) is the fraction of time that the transmitter is active (allowed to transmit) with a given window size, and hence provides a good indication of throughput attainable. Figure 5 shows this utilization as a function of window size for several values of \( RHO \) and assuming exponentially distributed service times (high variance). Realistic values of \( RHO \) in a typical PSN are typically about 0.1 since roundtrip delays are an order of magnitude larger than Host-to-network transmission times. Throughput rises linearly with window size (note the logarithmic scale for window size) up to the half-way point, and then somewhat more slowly. For constant service times (no variance), throughput would rise linearly with window size all the way to its maximum value, while service time distributions (roundtrip delays) with intermediate variance would behave between these limits.
This model provides information for both flow control concerns expressed above. To achieve maximum throughput, a window size approximately equal to the roundtrip delay divided by the Host-to-network transmission time for a packet is optimal (a larger window does not increase throughput much). This confirms the intuitive approach of "keeping the pipe full" between sender and receiver. To limit throughput to a desired fraction of the bandwidth available, the same fraction of this optimal window size should be used.

As long as packet loss and damage probabilities remain low, the above analysis shows how throughput may be flow control limited by a small window size. In networks where transmission errors are more likely, throughput may become retransmission limited because retransmissions of pending packets have priority over new transmissions. In general, achievable throughput will be the minimum due to either flow control or retransmission constraints.

A third similar constraint may be imposed by the source further restricting the window size for a connection (below what the receiver might allow) in order to share its communication resources fairly among all connections. For example, the ARPANET TIP limits each terminal (up to 300 baud) to a window of 6-12 characters in order to share its relatively scarce buffer space among all users. In the next section, the effects of destination buffer space limits or protocol performance will be explored.

DESTINATION BUFFERING AND ACKNOWLEDGMENT STRATEGIES

Buffering of received packets at the destination is necessary to smooth uneven production and consumption rates (hold packets until they can be processed) and to hold out-of-order arrivals for proper sequencing (if the protocol provides a sequencing function). Inadequate space for either purpose means that successfully received packets may have to be discarded if no space is available to hold them, increasing retransmission rate and delay.

In an attempt to avoid these problems, "conservative" acknowledgment strategies delay returning permission to send new data (c.f. last section) until buffer space to receive it has actually been made available by "consuming" arrived packets or furnishing additional space. In this case, no arriving packets need be discarded, but roundtrip delay for flow control credits is increased because processing time is included, and hence throughput may be reduced as shown in the last section.

Under favorable conditions, new flow control credits can be returned immediately on successful receipt of a packet, reducing the roundtrip time. As long as packet arrivals and receiver processing proceed smoothly (at regular intervals), no packets will be discarded. A larger window size than the buffer space available may be used to achieve higher throughput with reduced buffering requirements. Storage space along the network transmission path actually provides the rest of the space in the window.

Another simple queuing model provides insight into the performance of this optimistic strategy. The receiver is represented by a single server with mean service (processing) rate $u$, and size-limited queue of $N_{buf}$ packet buffers. Packets arrive at a mean rate $A$ determined by retransmission and flow control constraints discussed earlier, and the ratio of arrival rate to processing rate is $RHO = A/u$. Assuming exponential distributions, well-known queuing results give the
probability that an arriving packet will find all the buffers full, and have to be discarded.

\[
\text{P\text{full}}(\text{Nbuf}, \text{RHO}) = (1 - \text{RHO}) \cdot \frac{\text{RHO}^{\text{Nbuf}}}{1 - \text{RHO}^{\text{Nbuf}+1}}
\]

Three general cases may be distinguished.

For \( \text{RHO} >> 1 \), (arrival rate >> processing rate), nearly all arriving packets will be discarded regardless of the size of \( \text{Nbuf} \). In this case, throughput is clearly receiver rate limited, and the window size should be reduced to limit the sender's transmission rate to the receiver's processing rate.

For \( \text{RHO} << 1 \), very few packets will be discarded even if \( \text{Nbuf} \) is small. A fast receiver requires very little buffering, and buffer pooling among multiple connections may be advantageous.

For \( \text{RHO} = 1 \), (matched sending and receiving rates), the number of buffers required depends heavily on the "smoothness" of source and destination activity. Very regular activity allows a minimum of buffering. Unfortunately, irregular packet arrival and processing rates are more typical in computer networks, particularly in multiprogramming systems where process activity occurs in bursts. If process scheduling intervals in multiprogramming systems are large compared to roundtrip delays, then large window sizes and buffer allocations may be necessary to achieve high throughput.

**PACKET SIZE**

In transmitting large amounts of information between processes, a communication protocol must determine the amount of data to transmit in each packet. The primary result of varying packet size is to vary transmission delay through the network, \( f(t) \). Transmission delay in a store-and-forward network has a large component proportional to packet length because the transmission time on each hop between switching nodes is equal to packet length divided by bandwidth (although some networks either fragment or combine submitted packets before internal transmission).

Assuming this proportionality holds, shorter packets mean lower per packet delay, with ensuing effects on protocol performance as described in earlier sections. Unfortunately, overhead for short packets increases since each packet carries a fixed amount of header and control information, and more acknowledgments and general processing for the same amount of data will be necessary. Hence, maximum throughput attainable decreases while line efficiency (and total cost in bits transmitted) increases for shorter packets. Longer packets reduce overhead and allow higher throughput at the cost of increased delay. Packet switching networks typically impose an upper bound on the size of packets submitted for transmission in order to manage their internal resources and to ensure prompt access to other customers.

To illustrate some of these trade-offs, Figure 6 shows the total time to transmit a large block of information using different packet sizes, assuming typical PSN characteristics (but no packet loss) and transmission delay proportional to packet length. Using the smallest packets, throughput is so low that total delay is high. For very large packets, the increasing delay per packet also leads to high total delay. The "optimal" value for this application results from an intermediate packet size.

Other applications may have other priorities for cost, throughput, and delay performance. Transaction or interactive applications may select short packets to achieve low delay at somewhat higher cost and reduced throughput. Minimum cost or maximum throughput users willing to tolerate larger average delays may use long packets. "Real-time" traffic requiring moderate delay and good throughput for moderate block lengths may use intermediate packet sizes. As network bandwidth increases and error rates drop, the impact of packet length on protocol performance should lessen.

**CONCLUSIONS**

Distributed processing environments such as computer networks demand new techniques for providing reliable and efficient communication between remote processes. Good performance of communication protocols developed for this purpose depends heavily on tuning protocol parameters to complement network transmission characteristics. This paper has discussed the impact on communication efficiency of major protocol policies concerning retransmission, flow control, buffer allocation, acknowledgment, and packet size.
More complete results and more detailed development of the models underlying these results are contained in another paper by the author. Experiments are currently under way to verify the predictions of these models using the newly developed Transmission Control Program as an interprocess communication protocol.

ACKNOWLEDGMENTS

The author wishes to thank V. Cerf, Y. Dalal, R. Metcalfe, and S. Gaines, for their helpful discussion of the issues and review of the manuscript. Personnel and facilities at several ARPANET sites were also instrumental in supporting this work.

REFERENCES


Performance of file directory systems for data bases in star and distributed networks*

by WESLEY W. CHU
University of California, Los Angeles
Los Angeles, California

ABSTRACT

Three classes of file directories for distributed data bases are studied: The centralized directory system, the local directory system, and the distributed directory system. The parameters considered are communication cost, storage cost, code translation cost, query rate, update rate, directory size, and directory response time. This study reports the cost performance tradeoffs of these three classes of directory systems in star and distributed networks and provides a guide in the design of file directory systems when operating in these network environments.

INTRODUCTION

In the automation of large information systems, a major portion of the planning is concerned with methods of storing, updating, retrieving, and distributing large quantities of information in an information processing system. Examples of such efforts are found in business, medicine, library and management information systems. These systems, which may consist of several geographically separate divisions, need to process information files in common and thus form a network information processing system known as a distributed data base. One of the significant advantages of this network environment is the resource sharing capability. Problems that arise in the design of distributed data bases include file allocation policy,1 avoidance of file deadlock,1 file directory design, file partitioning policy, file reliability, privacy, and security issues, and interfacing considerations. This paper concerns itself with the allocation of directories in a distributed data base.

A directory is a listing of files available to the users of the network. Such a directory will enable a user at any node to determine where in a network a specific sharable file exists. One can consider such a directory to be similar to a card catalog of a public library.

These sharable files are referred to as public files. Users at each node may offer to list their files in this directory of public files for sharing purposes. A user may interrogate this list to determine its contents or to obtain information on where a specific sharable file exists. The list of non-shared files is assumed to be stored at the computer that is known to its users and, therefore, is not considered here. Further, we assume that each computer has its own local directory which consists of all the public files stored in that computer. To locate a file that is not in the local computer, the user must consult the file directory.

There are several ways to design the file directory: centralized file directory, local file directory and distributed file directory. Based on the computer network topology, operating cost (communication cost, storage cost, and code translation cost), directory query rate and directory update rate, we use mathematical models to study the operating cost and response time of these directory systems as a function of directory query rate, directory update rate, and the ratio of storage cost to communication cost. Two numerical examples are given (one with a star network topology, the other with a distributed network topology) to illustrate the applications of the models. The results provide us with insights on how to optimally plan for a file directory for a distributed data base.

THE MODEL

In this section, several mathematical models are introduced to study the performance of file directory systems for operating in the star network and distributed network topologies. Let us first consider the centralized file directory system.

Centralized file directory system

Single master directory

In the centralized file directory system, a master directory is located at one of the computers. When a
user requires a file that is not stored at his local directory, he consults the master directory to find out the location or content of that file. The centralized directory must be updated when there is a change in the storage location or contents of a file, or a creation of a new version of the file. Directory updating in such a system is relatively easy. However, there are communication costs incurred for each transaction. The operating cost for such a system per unit time is

\[
C_c = \sum_{i=1}^{n} C_{idt} \sum_{j=1}^{m_i} q_{ij} (2 + p_i) + \sum_{i=1}^{n} \sum_{j=1}^{m_i} [T_{id} (2 + p_i)] + C_d F
\]

where:  
- \( m_i \) = total number of files in the \( i^{th} \) computer  
- \( n \) = total number of computers in the network  
- \( d \) = location of the master directory  
- \( C_{idt} \) = transmission cost per character and per unit distance from the \( i^{th} \) computer to the master directory  
- \( s_{id} \) = distance between the \( i^{th} \) computer to the master directory.  
- \( l_i \) = average record length in characters for each transaction at the \( i^{th} \) computer, \( i=1,2,\ldots,n \)  
- \( q_{ij} \) = directory query rate by the \( j^{th} \) file at the \( i^{th} \) computer per unit time. \( j=1,2,\ldots,m \).  
- \( p_i \) = normalized directory update rate (the ratio of directory update rate to directory query rate) of the \( j^{th} \) file generated by the \( i^{th} \) computer.  \( 0 < P_i \leq 1 \).  
- \( T_{id} \) = code translation cost per transaction by the \( i^{th} \) computer at the master directory.  
- \( C_d \) = storage cost per character per unit time at the master directory.  
- \( F \) = size of the master file directory in characters.

The first term in Equation (1) represents the transmission cost for querying and updating the file directory, the second term represents the code translation cost (due to non-uniform information representations) for transactions (querying and updating) performed by all the computers at the directory, and the last term is the cost for storing the master file directory. 

To simplify the notation in Equation (1), we let \( q_i = \sum_{j=1}^{m_i} q_{ij} \). The \( q_i \) is the total number of queries generated per unit time by the \( i^{th} \) computer for the directory. Further, we assume that all the files at the \( i^{th} \) computer have the same directory update; that is, \( P_i = P_j \) for all \( j \). Equation (1) then reduces to

\[
C_c = \sum_{i=1}^{n} C_{idt} s_{id} q_i (2 + p_i) + \sum_{i=1}^{n} T_{id} (2 + p_i) + C_d F
\]

Extended centralized directory

In a centralized file directory, once the user finds the location or description of a file, he can append this information onto his local directory. Should the user use this file again, the directory information for this file can be obtained from his local directory, thereby reducing the communication cost as well as time for querying the master directory. However, when the information of that file at the master directory is updated, we also require updating of the information of that file in the local directories. Therefore, for notification of future updates in the master directory, the list of local directories that have appended file information is recorded in the master directory. The operating cost of such a system is:

\[
C_{EC} = \sum_{i=1}^{n} C_{idt} s_{id} q_i p_i + \sum_{i=1}^{n} q_i P_i T_i + C_d (F + F_e) + \sum_{i=1}^{n} a_i C_i + \sum_{i=1}^{n} \sum_{k=1}^{n} q_{ik} P_{ik} T_{ik} - C_d S (F + F_e) - \sum_{i=1}^{n} \sum_{k=1}^{n} a_i C_i
\]

where \( a_i \) = fraction of the master directory being appended at the local directory of the \( i^{th} \) computer, \( 0 \leq a_i \leq 1 \) \( i=1,2,\ldots,n \).  
- \( \beta_{ik} \) = probability that a file at the \( k^{th} \) computer has a transaction at the \( i^{th} \) computer, \( 0 \leq \beta_{ik} \leq 1 \) \( i=1,2,\ldots,n \) and \( k=1,2,\ldots,n \).  
- \( T_{ik} \) = translation cost for the master directory to perform a transaction with the \( k^{th} \) computer.  
- \( F_e \) = the size of the list of all the files stored at the master directory that requires notification for directory updates (in characters).

We assume that compared with the overall operating cost, the first cost of locating a file in the master directory is negligible. The first term in Equation (3) is the transmission cost for updating the master directory, the second term is the code translation cost for updating the master directory, the third term is the cost for storing the directory and the list of the transaction records for notification of future file directory updates, the fourth term is the cost for storing the extended file directories at all the local directories, and the last term is the communication cost and translation cost for the master directory to update the relevant appended local directories.

Multiple master directories

When the computers in a network are clustered in groups, it is often cost effective to provide a master directory at each of these clusters. The savings in...
communication cost for a multiple master directory system could far outweigh the cost of storing and updating the file directories. We can partition the n computers in the system into r clusters (r ≤ n) such that
\[ \sum_{j=1}^{r} n_j = n \]
where n_j is the number of computers in the jth cluster.

During normal operation, the computers in the jth cluster will query the jth directory which is a member of the jth cluster. One way to partition the n computers into r groups is to base on the network topology such that the partitioned clusters yield minimum communication costs. Another way is to base the clusters on directory query rate and directory update rate to achieve minimum response time. The operating cost for such a system is:
\[ C_m = \sum_{k=1}^{r} \left( \sum_{j=1}^{r} C_{o_k}\delta_{o_k} l_j q_i(2+p_i) + \sum_{j=1}^{r} T_{d_j}(2+p_j) \right) \]
\[ + \sum_{k=1}^{r} C_{d_k}^s F_i \sum_{i=1}^{r} \sum_{j=1}^{r} \left( C_{o_k}\delta_{o_k}+\delta_{o_k}\alpha \cdot l_i+T_{o_k}\delta_{o_k} \right) p_i d_i \]  
(4)

where r = number of master directories in the system,
\[ d_k \text{ is the kth master directory, } k=1, 2, \ldots, r. \]
\[ (i)e_d_k \text{ is the set of computers that uses the kth master directory} \]
\[ C_{o_k}\delta_{o_k} \text{ is storage cost per character per unit time of the kth master directory} \]
\[ C_{o_k}\alpha \text{ is transmission cost per character per unit distance from the i th computer to the kth master directory, } i=1, 2, \ldots, n \]

The last term of (4) is the communication cost and translation cost for updating the rest of the multiple directories in the system. For the single master directory case (r = 1), Equation (4) reduces to (2).

**Local file directory system**

In the local directory case, there is no master directory in the system. When a requested file is not stored in the user's local directory, the user queries all the other local directories in the system until the requested file has been located. Such a directory system requires high communication cost and translation cost, as well as search time for locating the file. For a system of n computers, it requires an average of (n-1)/2 directory queries to locate a file. Assuming the directory can only be updated by its owner, updating is done at its local directory which does not require communications cost. The operating cost for such a system is:
\[ C_L = \frac{1}{2} \left[ 2 \sum_{i=1}^{n} \sum_{j=1}^{n} C_{a_i}\delta_{a_i} l_j q_j + \sum_{i=1}^{n} T_{n_i} \sum_{j=1}^{n} q_j \right] \]  
(5)

where T_{n_i} is the code translation cost per transaction for the i th computer at the kth computer. The first term of Equation (5) is the expected communication costs for querying to and replying from all the local directories for those files that are not stored in the local computer, and the second term is the expected code translation costs for transactions with all the local directories in the system. The factor 1/2 is for taking the average operating cost. Since there is no master directory in such a system, there is no storage cost in Equation (5).

For simplicity in notation, we let q_i = \sum_{j=1}^{n} q_j. Eq. (5) becomes
\[ C_L = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{a_i}s_{a_i} l_j q_j + T_{n_i} q_j + \sum_{i=1}^{n} C_{a_i}s_{a_i} l_q \]  
(6)

Let us consider the case where each of the computers contains a routing table which routes the directory query directly to the other computers rather than returning the negative query reply to the sender. As a result, the expected total communication cost can be greatly reduced, particularly if the routing sequence takes into consideration the probability of finding the file in the directory. The total operating cost reduces to
\[ C_L' = \gamma \sum_{i=1}^{n} \sum_{j=1}^{n} C_{a_i}s_{a_i} l_j q_j + T_{n_i} q_j + \sum_{i=1}^{n} C_{a_i}s_{a_i} l_q \]  
(7)

where 0 < \gamma < 0.5. The value of \gamma depends on the network topology and the policy used in the routing table. The last term of Eq. (7) is the communication cost for replying to the queries.

**Distributed file directory system**

In the distributed directory case, each computer in the system has a master directory. The advantage of this system is its fast response time. The disadvantage of this system is the cost of storing master file directories at each computer as well as the communication cost for updating all these directories. The operating cost per unit time of such a system is:
\[ C_D = \sum_{i=1}^{n} \sum_{k=1}^{r} C_{a_i}s_{a_i} l_j q_j \sum_{j=1}^{n} q_j p_j \]  
\[ + \sum_{i=1}^{n} \sum_{j=1}^{n} T_{n_i} \sum_{j=1}^{n} q_j p_j + \sum_{i=1}^{n} C_{a_i} F \]  
(8)

where

- Communication Cost
- Translation Cost
- Storage Cost

The first term of Equation (8) is the communication cost for updating all the distributed master direc-
tories in the system, the second term is the translation cost associated with the updating of all the distributed directories, and the last term is the cost for storing all the master directories. If we assume that the directory update rates of all the files at the ith computer are identical, then $P_{ij} = P_i$. Again for simplicity in notation, we let $q_i = \sum_{j=1}^{m_i} q_{ij}$. Then (8) reduces to

$$C_D = \sum_{i=1}^{n} \sum_{k=1}^{n} (C_{ik} s_{ik} + T_{ik}) p_{ik} + \sum_{i=1}^{n} C_i F$$  \hspace{1cm} (9)$$

**OPERATING COST TRADEOFFS OF DIRECTORY SYSTEMS**

The intersection of two operating cost curves $C_x(P)$ and $C_y(P)$ (Figures 1 and 2) represent the cost tradeoff point (in terms of update rate) for directory systems x and y. If we assume that all the computers in the system have identical directory update rates, then the operating cost is a linear function of the normalized update rate $P$. Thus $C_x(P)$ and $C_y(P)$ can be expressed as:

$$C_x(P) = a_x P + b_x$$

and

$$C_y(P) = a_y P + b_y$$

where $a_x$ and $a_y$ are incremental costs for directory updates and $b_x$ and $b_y$ are fixed directory operating costs. The intersection point of $C_x(p)$ and $C_y(P)$, $P(x,y)$, satisfies

$$P(x,y) = \frac{b_y - b_x}{a_x - a_y}$$  \hspace{1cm} (10)$$

Let us now consider the intersection of the cost curves for the centralized and extended centralized directory systems. From Equation (2), we have

$$a_x = \sum_{i=1}^{n} C_{id} s_{id} q_i + \sum_{i=1}^{n} T_{id} q_i$$

and

$$b_x = \sum_{i=1}^{n} 2C_{id} s_{id} q_i + \sum_{i=1}^{n} 2T_{id} q_i + C_x F$$

From (3), we have

$$a_{EC} = \sum_{i=1}^{n} C_{id} s_{id} q_i + \sum_{i=1}^{n} T_{id} q_i + \sum_{i=1}^{n} \sum_{k=1}^{n} q_{ik} (C_{ik} s_{ik} + T_{ik})$$

and

$$b_{EC} = C_x (P + F_x) + \sum_{i=1}^{n} a_i C_i F$$

Substituting $a_x$, $a_{EC}$, $b_x$, and $b_{EC}$ into (10) and simplifying, we have

$$P(c,EC) = \frac{\sum_{i=1}^{n} q_{ik} (C_{ik} s_{ik} + T_{ik})}{\sum_{i=1}^{n} n_{id} q_{ik} (C_{id} s_{id} + T_{id})} \hspace{1cm} (11)$$

**DISTANCES IN MILES**

Figure 1 — Performance of directory systems for a star network

$C/C^* = 10$ month/mile—(a) A star network
In order to simplify equation (11), we assume that
(1) the communication cost is much higher than the
storage cost so that the storage cost becomes negligible,
(2) all the computers in the system are using the same
software code, thus translation cost is not required;
that is, $T_i = 0$, and (3) $\beta_i = 1$, $l_i = 1$, and $q_i = q$ for all $i$,
then (11) reduces to

$$P(c, EC) \approx \frac{\beta}{n-1}$$

(12)

For example, if $n = 10$ and $\beta = 1/3$, then $P(c, EC) = 2/3 = 0.667$. Thus for a network with ten computers op­
erated in the above stated environment, when the di­
rectory update rate of each computer is less than 67
percent of its query rate, the extended centralized
directory system yields a lower operating cost than
that of the centralized directory system.

We shall now consider the directory operating cost
tradeoff between the centralized directory system and
the distributed directory system. From (9), we have

$$a_D = \sum_{i=1}^{n} \sum_{k=1}^{n} (C_{ik} s_{ik} l_i + T_{ik}) q_i$$

$$b_D = \sum_{i=1}^{n} C_{i} F$$

Compared with the communication cost, the storage
cost again can be assumed to be negligible. Further we

$$P(c, D) = \frac{\sum_{i=1}^{n} \sum_{k=1}^{n} (C_{ik} s_{ik} l_i + T_{ik}) q_i}{\sum_{i=1}^{n} \sum_{k=1}^{n} C_{ik} F}$$

(13)
3.5

UNIFORM QUERY RATE

NON UNIFORM QUERY RATE

3.0

2.5

2.0

1.5

1.0

0.5

0

0

0.2

0.4

0.6

0.8

1.0

UPDATE RATE/QUERY RATE

OPERATING COST (10^8$/MONTH)

assume that \( l_i = 1, q_i = q, \) for \( i = 1, 2, \ldots, n; \) \( C_{ik} s_{ik} = C_{ik} s_{ik} \) and \( T_i = T_{ik} = 0. \) Then (13) reduces to

\[
P(C, D) = \frac{2}{n - 1} \quad (14)
\]

For a network with ten computers operated in the above stated environment, from (14) we know that \( P(C, D) = 0.22 \) which implies that when the directory update rate is less than 22 percent of its query rate, the distributed file directory system yields a lower operating cost than that of the local file directory system.

We shall now consider the intersection of the local file directory cost curve \( C_i(P) \) with the distributed file directory cost curve \( C_D(P) \). From (6), we have

\[
a_i = 0
\]

\[
b_i = \sum_{k=1}^{n} \sum_{l=1}^{n} [C_{lk} s_{lk} q_l + T_{lk} q_k]
\]

Substituting \( a_i, a_n, b_i, \) and \( b_n \) into (10), we have

\[
P(C, D) = \frac{- \sum_{i=1}^{n} C_i + \sum_{k=1}^{n} (C_{ik} s_{ik} l_i + T_{ik}) q_i}{\sum_{i=1}^{n} \sum_{k=1}^{n} (C_{ik} s_{ik} l_i + T_{ik}) q_i} \quad (15)
\]

When \( C_{ik} s_{ik} \gg C_{ik}, P(C, D) \rightarrow 1. \) This implies that when the communication cost is high as compared to the storage cost and when the directory update rate is less than the directory query rate, the distributed file directory system yields a lower operating cost than that of the local file directory system.

Comparing the approximate results obtained from Equations (12), (14), and (15) with those from direct computation as shown in Figure 3, we note that they agree quite well at high communication cost. Therefore (12), (14), and (15) may be used to estimate the approximate operating cost tradeoffs for high communication cost cases.

Figure 3—Performance of multiple directory systems for the distributed network. \( C_i/C = 10 \) month/mile, and query rate = 1000 queries/month. For \( r = 2, \) master directories at computers 1 and 2; For \( r = 3, \) master directories at computers 1, 2, and 3.
DIRECTORY QUERY RESPONSE TIME

In this section, we shall consider the query response time for various directory systems. The expected response time for the \(i^{th}\) computer to query its directory is defined as from initiation of a query at the \(i^{th}\) computer to the directory until the start of its reception. The expected response time consists of the waiting time at the input queue of the directory for processing the query \(t_{i}(i)\), the waiting time at the output queue of the directory for transmission \(t_{t}(i)\), the time to transmit the query to and its reply from the directory \(t_{s}(i)\), and the directory processing time \(t_{d}(i)\). The processing time consists of code translation, searching, and accessing. It depends on the file structure of the directory as well as the access time of the storage device and could be different from one system to another and should be known to the users. Here, therefore, we only consider the delay incurred at the input queue and output queue(s) of the directory, and the time to transmit the query on the communication channel. Let us denote the sum of these components as \(t_{a,d}\), known as the directory query response time from the \(i^{th}\) computer to the directory; thus

\[
t_{a,d} = t_{i}(i) + t_{s}(i) + t_{t}(i) + t_{d}(i) \quad (16)
\]

Clearly, the real query response time incurred by the users, \(t_{a,d}\), is equal to the sum of \(t_{a,d}\) and \(t_{r}(i)\).

The arrivals at the input queue of the directory are the queries and updates generated by all the computers. The arrivals at the output queue(s) of the directory are the query replies generated at the directory. We shall assume these query arrivals can be approximated by a Poisson Process. The time between interrupts by the computer to process directory queries is \(1/\mu\), where \(\mu\) is the communication line transmits at a constant rate of \(R\) characters/second, the time to transmit (send or reply) a query of 1 character is \(1/\mu=1/R\). Under these conditions, the waiting time can be computed from known queuing theory results. The average waiting time for a single server queuing system with Poisson arrivals and constant service time, \(M/D/1\), is

\[
W = \frac{1}{\mu} \cdot \frac{\lambda}{2(\mu - \lambda)}
\]

where

\[
1/\mu = \text{time to service a query in seconds} \\
\lambda = \text{average number of queries arrived/second}
\]

Assuming that queries and their replies generated at the directory input and output queues can be approximated as a Poisson process, then the query response time equals

\[
t_{a,d} = t_{i}(i) + t_{s}(i) = \frac{\lambda_{i}}{2\mu_{i}(\mu_{i} - \lambda_{i})} + \frac{\lambda_{d}}{2\mu_{d}(\mu_{d} - \lambda_{d})} + 2/\mu
\]

(18)

Let us first consider the centralized directory system. The total number of queries arriving at the directory, \(\lambda_{d}\), consists of directory queries and their updates from all the computers in the system. Thus \(\lambda_{d} = \lambda_{d} + \sum_{i=1}^{n} q_{i}(1+p_{i})\). Since the directory does not have to reply to update traffic, and since each destination has its own output queue, the arrival rate at the output queue for the \(i^{th}\) computer is equal to \(\lambda_{i} = \lambda_{d} = q_{i}\). By substituting \(\lambda_{i}\) and \(\lambda_{d}\) into (18), we have

\[
t_{a,d} = \frac{\sum_{i=1}^{n} q_{i}(1+p_{i})}{2\mu_{i}(\mu_{i} - \lambda_{i})} + \frac{\lambda_{d}}{2\mu_{d}(\mu_{d} - \lambda_{d})} + 2/\mu
\]

(19)

For the computer that stores the master directory, \(t_{a,d} = t_{i}(i) = 0\). The expected query response time \(t_{a,d} = t_{i}(i)\).

Let us now consider the extended centralized directory system. For those files that have not yet been queried at the directory by the \(i^{th}\) computer, the response time is similar to (19) except that the directory query rate is much smaller and the directory output queue for the \(i^{th}\) computer should also include the update traffic (generated by all the computers) for the \(i^{th}\) computer. For those files whose directory information has already been appended to the local directory at the \(i^{th}\) computer, there is no need for the \(i^{th}\) computer to consult the master directory about these files. Thus the query response time reduces to \(t_{a,d} = t_{i}(i)\).

For multiple master directory systems, since the directory queries are shared by the multiple master directories, the waiting time for processing the directory queries at each master directory is much lower than the single master directory system. Therefore, the response time for the multiple master directory is lower than that of the single master directory system. The query arrival rate at the \(k^{th}\) master directory \(d_{k}\) is

\[
\lambda_{i} = \sum_{i=1}^{n} q_{i} + \sum_{i=1}^{n} p_{i} q_{i}
\]

and the query reply rate at the output queue for the \(i^{th}\) computer is \(\lambda_{d} = q_{i}\). Thus the response time for the \(i^{th}\) computer to the \(k^{th}\) master directory can be computed from (19) by replacing the \(\sum_{i=1}^{n} q_{i}(1+p_{i})\) in the first term of (19) by \(\sum_{i=1}^{n} q_{i} + \sum_{i=1}^{n} p_{i} q_{i}\). When directory queries are generated by those computers that store the master directories, these query replies do not require transmission. Thus the response time equals \(t_{i}(i)\).

For the local directory system whose replies are returned directly to the sender (i.e., without routing), the \(i^{th}\) computer may locate the information of the file before reaching the \(k^{th}\) local directory. Therefore on the average only half of the queries generated at the \(i^{th}\) computer will reach the \(k^{th}\) computer. Thus the query arrival rate at the input queue of the \(k^{th}\) local directory is \(\lambda_{i} = \frac{1}{2} \sum_{i=1}^{n} q_{i}\), where \(q_{i}\) = directory query rate.
generated by the \( i \)th computer. Assuming there is an output queue for each destination at the \( k \)th computer, then the arrival rate at the output designated for the \( i \)th computer is \( \lambda_i = q_i \). Substituting \( \lambda_1 \) and \( \lambda_2 \) into (18), we have

\[
T_{i,k} = \frac{\sum_{i=1}^{n} q_i}{\lambda_i} \left( \frac{1}{\mu_i} - \frac{1}{2} \frac{1}{\sum_{i=1}^{n} q_i} \right) + \frac{1}{2} \frac{l_{q_i}}{R} \left( \frac{2R}{1-q_i} \right) + \frac{2l}{R} \tag{20}
\]

Because the local directory system uses routing, queries are passed around from one directory to another rather than sent directly to the sender. With a carefully designed routing strategy, the input traffic rate \( \lambda \) could be greatly reduced. Further, the third term of (20) reduces to half of its value. Therefore, the query response time for the system with routing could be much smaller than that without routing.

Assuming all the files in the data base are equally likely to be stored in any local directory system, the time to locate a file by the \( i \)th computer is

\[
t_i = \frac{1}{2} \sum_{k=1}^{n} t_{i,k}
\]

and the real expected directory response time for the \( i \)th computer is

\[
t^* = \frac{1}{2} \sum_{i=1}^{n} \left( t_{i,k} + t_{i,k} \right)
\]

For the distributed directory system, each computer has a master directory. Public file information can be obtained from the file directory at the user's computer. Thus, \( t_{i}(i) = t_{i}(k) = 0 \). The query arrival rate at the directory input queue is \( \lambda_i = q_i \left( 1 + \sum_{k=1}^{n} p_k \right) \). The expected directory response time reduces to

\[
t_{i,k} = t_i(i) = q_i \left( 1 + \sum_{k=1}^{n} p_k \right) \left( \frac{1}{\mu_i} - q_i \left( 1 + \sum_{k=1}^{n} p_k \right) \right)
\]

NUMERICAL COMPUTATIONS

From the models we developed in the last section, we know that the operating cost of the directory system depends on many parameters such as network topology, transmission cost, storage cost, translation cost, directory query rate, directory update rate, and directory size. Although the models are formulated in general terms and can be applied to different parameter values, in order to draw some meaningful conclusions as well as limit the number of study cases, we shall select two types of network topologies: a star network which has the property that the distances from the center node to all the computers are equal, and a distributed network of computers that forms clusters, the distances among these clusters being unequal. In our example we assume that all the computers in the system have the same storage cost, code translation cost, and directory update rate; the size of the master file directory is directly proportional to the number of computers in the system; all communication channels have full duplex capabilities and the same communication cost. First, we study the star network as shown in Figure 1a. We computed the monthly operating cost as a function of file directory query rate and directory update rate for various directory systems. Numerical results are shown in Figures 1b and 1c. The parameters used in this example are: \( C/C=10 \) month/mile, \( T/C=2000 \) byte month/transaction, \( C^* = 7.0 \times 10^{-4} \$ \)/byte month, \( F = 6 \times 10^6 \) bytes, \( F_1 = 3 \times 10^4 \) bytes, \( l = 50 \) bytes, \( a = 0.1 \), \( \beta = 1/3 \), and \( q_i = 1000 \) queries/month. The directory update rate is expressed in terms of the normalized directory update rate which is the ratio of directory update rate to directory query rate.

In the same manner, we study the distributed network which consists of ten computers that forms into three clusters as shown in Figure 2(a). Using the same parameter values as for the star network, we compute the monthly operating cost as a function of directory query rate and normalized directory update rate. The results are given in Figures 2(b) and 2(c).

So far we have assumed that each computer has the same directory query rate. We shall now relax this assumption and let the query rate at each computer be different. In order to compare with the uniform query rate case, we let the total number of directory queries generated from all the computers remain the same but vary the query rates among the computers. For example, we let computers 1, 2, 3, 4, and 5 have a rate of 1500 queries/month and others 500 queries/month. The results reveal that the operating cost of the centralized directory system with the above stated query rate is lower than that of the case that 1000 queries/month generated from each computer (Figure 2(c)). On the other hand, if we interchange the directory query rate of computers 1 to 5 with computers 6 to 10, then the operating cost of the centralized directory system with the above non-uniform query rate is higher than that of the uniform query rate case.

Next, we study the operating cost characteristics of the multiple master directories system. According to the distance among computers, we partitioned the ten computers into three groups in which each computer belongs to one of the three master directories. We evaluate the monthly operating costs as a function of directory update rate from Equation (5) and compare them with those of the extended centralized directory system, the local file directory system, and the distributed directory system as shown in Figure 3.

Figure 4 shows the operating cost trade-offs for file directory systems as a function of the ratio of communication cost to storage cost. The results are obtained from the intersection of the operating cost curves un-
delay occurring at the directory input queues, directory query response time, and the traffic intensities with R=960 char/sec are portrayed in Figure 5.

**DISCUSSION OF RESULTS**

From the numerical examples studied in the last section we notice that the operating cost of the file directory depends greatly on the directory query rate and the directory update rate. Because of the large amount of data communication and translation associated with the directory updates in the distributed directory system, the rate of increase in operating cost with respect to directory update rate for the distributed directory system is higher than that of the centralized directory system. In the local directory system, we only need to update the local directory that generates the update and no transmission is required. Therefore, the operating cost is independent of the directory update rate. For a given network topology the operating costs of the local directory systems are higher than the centralized directory systems.

Assuming that the transmission cost is higher than the storage cost, our study reveals that when the directory update rate is low (e.g., less than 10% of the query rate), the distributed directory system yields lower operating costs than the centralized directory system. As directory update rate increases, the oper-

From the collection of the Computer History Museum (www.computerhistory.org)
Comparing the two types of centralized directory systems, the extended centralized directory yields lower operating costs than the centralized directory at low directory updating (less than 50% of the query rate), and the performance reverses at high directory update rates. This is because of the excessive data communications required in the extended centralized directory system to update all the extended local directories. The exact cross-over point of the operating cost curves for these two types of directory systems depends on network topology and such parameters as storage cost, transmission costs, translation costs, etc. As the directory update rate increases, the performance characteristics of the extended centralized directory system become similar to that of the distributed directory system.

We also studied the influence of the distribution of the directory query traffic on the operating cost. In order to provide a common base for comparison, we kept the total number of queries generated by the computers to be a constant, and varied the query traffic among the computers. We found that the traffic distribution does not have an effect on the directory operating cost when all the computers are equal distances from each other, and does have an effect on the operating cost when the distances among the computers are different.

When a network of computers forms in clusters (Figure 2(a)) and when their directory update rates are low, our studied example reveals that installing a master directory at each cluster requires less communication cost and therefore yields better performance than the extended centralized directory system as shown in Figure 3.

Figure 5(a) displays the queuing delay occurring at the input of the directory. The queuing delay increases as the query rate increases. Except in the local directory case, the queuing delay increases as the directory update rate increases. This is because the update messages are considered as input traffic to the directory. Since the input traffic to the multiple master directory system is shared among the master directories, it yields lower queuing delay than the centralized directory system. In the distributed directory system, the input traffic consists only of queries generated from its initiator and the directory updates generated from the rest of the computers in the system. Therefore such systems have the lowest delay at the directory input.
Performance of File Directory Systems

CONCLUSION

Based on the computer network topology, communication cost, storage cost, code translation cost, directory query rate, and directory update rate, we studied the cost-performance tradeoffs of the three classes of directory systems. Assuming that the transmission cost is much higher than the storage cost, our study reveals that for low directory update rates (less than 10% of the query rate), the distributed file directory yields a lower operating cost than the centralized directory system. When the update rate is greater than about 15% of the query rate, the centralized directory yields a lower operating cost than the distributed file directory system and also the local directory system. The extended centralized file directory system yields a lower operating cost than the centralized directory systems at low directory update rate (for example, less than 50% of the query rate). For a system which has a very low directory update, for example, less than 5-10% of query rate, the extended centralized directory system should be used. When a network topology forms in clusters and when directory updating is greater than 5-10% of the query rate, multiple directories systems with a master directory at each cluster yield a lower operating cost and directory response time than the extended centralized file directory system.

In the local directory system, because of the large amount of communication costs associated with searching a file when it is not stored in the user's local file directory, the operating cost is much higher than that of the centralized directory system. However, when the directory update rate is very high (greater than 50% of the query rate) and when communication cost is lower than the storage cost, the local directory system yields a lower operating cost than the distributed directory system. Since the local directory system requires a large amount of communications as well as search time for locating a file, efficient routing strategy could greatly reduce the operating cost and query response time in such systems.

We have also studied the cost tradeoffs when the storage cost is equal to or greater than the communication cost. Our study reveals that the cost curves are similar to the cases where the communication cost is higher than the storage cost except the intersection points of these cost curves differ as shown in Figure 4.

Since the distributed directory system and the extended centralized directory system (assuming that the requested file resides in the appended local directory at the user's computer) do not require communication for querying the directory, they yield lower file query response time than those of the centralized directory system and the localized directory system. When using the multiple master directories, the queries generated by the users are shared by the master directories in the system. Therefore, the directory query response time for the multiple directory system is lower than that of the single master directory system.

For a given network topology and operating environment, we can use the models developed in this paper to study the operating cost tradeoffs and directory response time for various directory systems. Such investigation provides us with a guide in designing directory systems for distributed data bases.

ACKNOWLEDGMENTS

The author wishes to thank E. E. Nahourai of IBM for his stimulating discussion in the formulation of the model and David Lee and Shaila V. Kinariwala of UCLA for their assistance in performing the computation.

REFERENCES

On measurement facilities in packet radio systems*

by FOUAD A. TOBAGI, STANLEY E. LIEBERSON and LEONARD KLEINROCK

University of California, Los Angeles, California

ABSTRACT

The growth of computer networks has proven both the need for and the success of resource sharing technology. A new resource sharing technique, utilizing broadcast channels, has been under development as a Packet Radio system and will shortly undergo testing. In this paper, we consider that Packet Radio system, and examine the measurement tasks necessary to support such important measurement goals as the validation of mathematical models, the evaluation of system protocols and the detection of design flaws. We describe the data necessary to measure the many aspects of network behavior, the tools needed to gather this data and the means of collecting it at a central location; all in a fashion consistent with the system protocols and hardware constraints, and with minimal impact on the system operation itself.

INTRODUCTION

This paper is primarily concerned with the unique measurement aspects of Packet Radio Systems as regards network evaluation, and considers the design of a set of measurement facilities, the development of data gathering techniques within the framework of the system design and the use of these measurements to evaluate the system performance and its operational algorithms.

The need for sharing of computer resources by organizing these resources into computer networks has been long recognized and the feasibility of constructing such networks has been demonstrated by many successfully operating network systems. Perhaps the most prominent example is the ARPANET, which utilizes the technique of packet-switching, appropriate for bursty computer network traffic, thus achieving better sharing of the communication resources.

The ARPANET emerged in 1969 as the first major packet-switching network experiment; since the essence of an experiment is measurement, and in line with Hamming's observation that "it is difficult to have a science without measurement", considerable care was taken from the beginning in the design and development effort to include the tools necessary and appropriate to satisfy the many measurement goals. As a result of well designed experiments on the ARPANET using these tools, valuable insight has been gained regarding the network usage and behavior.

The Packet Radio System is another yet different example of a computer resource sharing network. It is being developed by the Advanced Research Projects Agency in order to demonstrate the applicability of the packet radio concept in organizing computer resources into a computer communications network. It is this packet radio network which is of concern to us in this paper. The network is currently in its design phase, and, as was the case with the ARPANET, care is being taken to include the ability to measure network behavior. UCLA is in charge of this measurement effort.

This concern for measurement is due to several factors. Firstly, these measurements provide a means to evaluate the performance of the operational protocols employed and the identification of their key parameters. Moreover, this realistic observation of the system behavior will assist in the validation and improvement of existing analytical models devised to study some of these operational schemes, such as the access modes and routing strategies. Secondly, these measurements will allow for the detection of system inefficiencies and the identification of design flaws such as the inadvertent creation of a deadlock condition.

Thirdly, measurement facilities and data, when used to improve network design, are a valuable feedback process in which design deficiencies are detected and subsequently corrected. Wire networks differ from radio networks mainly in the omni-directional broadcast nature of the communication and consequently the protocols employed; therefore, it calls for new approaches in the design and implementation of the measurement facilities and their use.

* This work was supported by the Advanced Research Projects Agency of the Department of Defense (DAHC-15-73-C-0368).

* A preliminary demonstration of the system is under way. A prototype network is being set up in the Palo Alto, California, area.

589

From the collection of the Computer History Museum (www.computerhistory.org)
In the following section, we present an overview of the packet radio system concepts and a brief description of the currently specified operational procedures. In a later section, we describe the network measurement facilities which consist of the measurement tools and the techniques for data collection. In the last section, we identify and discuss in some detail the desirable measurement functions to satisfy the need for validation and performance evaluation outlined above.

THE PACKET RADIO SYSTEM

Several papers have already appeared in the literature which describe the packet radio concept and discuss many of the issues involved in the system design.1-3,9-10 In this section, we briefly describe these system components and operational procedures necessary to understand the measurement considerations presented below.

There are three basic functional components of a packet radio system:

(i) packet radio terminals—these are the sources and destinations of traffic on the packet radio network.
(ii) packet radio stations—these function as S/F switches for local traffic and as interfaces between the broadcast system and other computers or networks. Also, they perform directory, monitoring and control functions for the overall system, and they play a central role in that all traffic passes through the station, i.e., we have a centralized network.
(iii) packet radio repeaters—their function is to extend the effective range of terminals and stations by acting as Store-and-Forward relays.

The repeater, which has been developed by Collins Radio and is called a packet radio unit (PRU), consists of a radio transceiver and a microprocessor. The function of the PRU is to receive and transmit packets according to dynamic routing and control algorithms specified by the station. For simplicity and uniformity of design, the PRU is used as the front-end of terminal devices and of stations, interfacing them with the radio net. In Figure 1 we show an oversimplified picture of the PRU identifying its various sections: the radio transceiver, the store-and-forward software, the control process, and the measurement process.

In this initial system, the terminals, stations and repeaters are linked together by a single broadcast channel using omni-directional antennas. The repeaters do not determine routes. All the routing computations are performed by the station. A hierarchical routing algorithm is used which makes the routing in the broadcast network resemble routing in a point-to-point network by forming a hierarchical tree structure. This structure is constructed by having the station assign to each repeater a label which defines its position in the tree. A packet is routed along the path determined by the tree, requiring the packet header to contain a string of appropriate repeater ID's, or labels. Thus, neighboring repeaters hearing the broadcasted packet but not on the determined path will reject the packet rather than relay it. However, this algorithm is flexible in that it allows the repeater to seek an alternate route for a packet when a path seems to be blocked. Moreover, the station with its monitoring procedures can dynamically restructure the tree by re-labeling any of the repeaters in response to component failures or traffic congestion.

In order to achieve reliable packet transport, acknowledgment procedures are required. There are two types of acknowledgments: the end-to-end ac-
knowledgments (FTE) between end devices, and hop-by-hop acknowledgments (HBH) between repeaters. Except for the last hop on a packet’s route, HBH acknowledgments are passive in that the relaying of a packet over a hop constitutes an acknowledgment of the transmission over the previous hop; this “echo acknowledgment” is due to the omni-directional broadcast property. At the last hop, an active HBH acknowledgment must be generated.

MEASUREMENT FACILITIES

Several factors exist in the packet radio system which do not allow for a simple transfer of ARPANET-like measurement facilities to a packet radio network. Although the latter utilizes the same technique of packet-switching, the packet radio concept is unique in the constraints it places on all system operations and the measurement effort in particular.

The radio broadcast nature of transmissions is such that the transmission of measurement data not only introduces overhead over its own path, but causes transmission interference at neighboring repeaters within hearing distances and creates additional overhead on those PRU’s activities. Moreover, the desire to keep the components small and portable, as well as the limited speed of the IMP’s CPU within the PRUs, place significant constraints on the measurement facilities and their usage. The available storage is extremely limited and the overhead placed on the PRU’s CPU is of utmost importance in evaluating the feasibility of a measurement tool and of the collection of data in support of a measurement function. As the operational protocols of the net are different from wire nets, the measurement functions devised to support the evaluation of their performance are unique. Thus, the measurement effort consisted of identifying the measurement functions (as described in the following section) and devising the measurement facilities required to support those functions under the constraints that the system imposes. The development of the tools was an iterative design process seeking a balance between supporting the measurement functions and satisfying the system constraints, as well as making sure that the network communication protocols allow the implementation and proper functioning of those tools.

In this section, we describe the various types of statistics desired in the Packet Radio Net,* the traffic sources required in measurement experiments and the techniques available for measurement data collection. We shall postpone until the next section the detailed list of the quantities that will be measured by each of the types of statistics (tools).

Cumulative statistics (Cumstats)

As its name suggests these consist of data regarding a variety of events, accumulated over a given period of time, and provided in the form of sums, frequencies and histograms. We shall distinguish between those data collected at the PRUs (PRU based Cumstats) and those collected at the end devices (the end-to-end Cumstats). The PRU based Cumstats provide information about the local environment and behavior such as traffic load, channel access, routing performance, and repeater activity. Conversely, end-to-end statistics collected at network sources and sinks, that is stations and terminal devices, will reflect more global network behavior such as user delays and network throughput.

Trace statistics

The trace capability allows one to literally follow a packet through the network, and to trace the route which it takes and the delays which it encounters at each hop. In the ARPANET, selected IMPs gather data on packets to be traced (which may include any packet) and send this data to the collection point as a new packet. In the packet radio network, however, the collection of trace data at the repeaters is prohibited by the limited size of storage in the PRU. To overcome this problem, we have introduced a new type of packet called the Pickup Packet.* These packets are generated with an empty text field by traffic generators at end devices. As these packets flow normally in the network according to the transport protocols, selected repeaters will gather the trace statistics and will store them within the text field of the pickup packets themselves.

Snapshot statistics

Snapshots give an instantaneous peek at a PRU, showing its state at that moment with regard to buffer assignment and queue lengths. (In the ARPANET, which is a decentralized network in which each node contains routing algorithms and data, snapshots also include routing related information; in the Packet Radio Network, such information is available at the station). Changes to appropriate station tables will be time stamped and collected as the station’s snapshot function.

Artificial traffic generators

Traffic sources

The creation of streams of packets between given points in the net, with given durations, intervals,

* These types of statistics, as well as traffic generators, which have been widely used in ARPANET measurement experiments, will differ significantly from those of the Packet Radio Network in regards to the specific quantities gathered and the means of collecting them at a central location.

* The notion of the pickup packet was first suggested by H. Oderbeck.
packet lengths, and packet types (Information and Pickup Packets) is clearly a requirement of any experimental system. While it might be desirable to provide each PRU with the capability of creating such traffic, this additional burden on the PRU software can be avoided if there exists a reasonable number of terminals with processors attached which, along with the station, will be programmed to provide the traffic-source functions indicated above.

Specifically, traffic-source features which the terminals (and Station) should provide are: (1) Information Packets—the user specifies the packet length, frequency, destination and duration of one or more streams of Information Packets. (The text content may be arbitrary.) (2) Pickup Packets—the user specifies the packet length, frequency, destination and duration of one or more streams of Pickup Packets.

In the initial system, there will be a limited number of system elements, making it desirable to simulate in a terminal a multi-terminal environment. That is, the traffic generated at a single terminal will emulate the traffic that would be generated by several separate sources. A great deal of complexity is introduced in the design of these devices because of the hardware and software capabilities required to support this function. Feasibility and techniques of achieving this is under investigation.

Station measurement process

Since the station is the central node and provides central control for the operation of the entire network it therefore plays a central role in the execution of measurement functions. It is through the station that the initiation and termination of measurement experiments is controlled. In particular, the station enables and disables the Cumstat and Pickup packets functions at the PRU’s, and assigns to the various elements the intervals for Cumstat collections, and to the artificial traffic generator, their corresponding parameters. Moreover, it is to the station that all measurement data is ultimately destined; upon arrival at the station, the data is time-stamped and stored in a single measurement file for off-line reduction and analysis. In addition, all changes to the station’s internal tables (routing, connectivity, PRU operational parameters, etc.) will be reflected by an entry into the measurement file, thus allowing the correlation of measurement results to the actual network configuration. A (measurement) process at the station will perform all of the above functions.

Measurement data collection

As mentioned earlier, pickup packets are generated at stations and terminals. Those packets generated at a terminal are destined to (and collected at) the station; those generated by the station will be returned by their destinations to the station as regular packets for collection into the measurement file.

Let us now discuss the techniques for centrally collecting cumulative statistics. The data, generated at the PRU’s or terminal devices, must be transmitted to the station using the PR Net itself. One way of achieving this is to form at the PRU, at the end of each Cumstat interval, a measurement packet called the Cumstat packet, which is time-stamped and transmitted to the station. The second method consists of having the station send at regular intervals to appropriate PRU’s an executable control packet* called an Examine packet which collects time stamped Cumstat data and which returns back to the station.

For purposes of analysis, it is desirable for the Cumstat data received at the station to correspond to equal length time intervals at the generating device. This can be achieved in the automatic method if reliable ETE transmission exists, i.e., if ETE acknowledgment capabilities are provided in all terminal devices and PRU’s, preventing the loss of a Cumstat packet from a device on its way to the station. In the absence of the ETE capability in the PRU’s, one may decrease the Cumstat intervals (thus increasing the frequency of transmitting Cumstat data), thereby decreasing the gaps between correctly received Cumstat packets. With the Examine method, variable length Cumstat intervals will occur since Examine packets, sent at regular intervals from the station, are subject to (i) the network random delays en route to the destination PRU, and (ii) the possibility of loss in either direction.

The choice of a collection method will have to take into consideration the overhead that it imposes on the PRUs and on the network.

MEASUREMENT FUNCTIONS

We have described in the previous section the measurement facilities that are desirable in a PRNET to support the measurement functions. In this section we shall identify and discuss these functions in some detail, determine the required data items and describe the role of these measurement facilities in supplying the data. These include: channel access, operational protocols, repeater performance, traffic characteristics, and the network’s global performance.

Channel access

One of the main features that distinguish the Packet Radio Network from point-to-point networks is that

* An executable control packet is a packet that originates at the station and is destined to a PRU. It contains code to be executed by the destination PRU. In particular, the Examine packet contains the necessary code to load the contents of specified memory locations into the text of the packet for shipment back to the station.
devices transmit packets over a broadcast channel by using a random access scheme. These random access schemes are characterized by the sharing of a single channel in a multi-access fashion, thus allowing for packet interference to occur. Considerable progress has been made in analyzing these access modes, which include pure and slotted ALOHA and the more recently developed techniques of Carrier Sense Multiple Access (CSMA). In the initial experimental system pure ALOHA, 1-persistent CSMA and non-persistent CSMA will be available. Our measurement aims are to validate the analytical models of the three access modes and to evaluate their performance in realistic environments.

In evaluating terminal access in a single hop system (a model commonly used in analysis), we consider an environment consisting of a single station and a population of terminals within range and in line-of-sight of the station. In order to determine the relationships between the network throughput (rate of successfully received packets at the station) and channel traffic (rate of packet transmissions over the channel), as well as the relationships between the network throughput and packet delays, the following quantities will be measured:

(a) the number of transmissions a packet incurs before success
(b) the one-hop packet delay: time elapsed since the packet is ready for transmission until it is acknowledged, i.e., until its acknowledgment packet is received from the station
(c) network throughput: average number of packets received at the station per unit time

Items (a) and (b) are obtained in the form of histograms by the Cumstat tools at the PRD and the end device respectively. Item (b) may also be obtained individually for each Pickup packet by having the originating device store in it its time of generation and its transmission times, and in the succeeding Pickup packet, store the time its acknowledgment arrived. Item (c) is obtained at the station from end-to-end cumulative statistics.

The task of measuring performance of terminal access techniques in multi-repeater environments differs from the previous one in that repeater-to-repeater traffic is present contending on the same channel. The environment consists of a number of repeaters and stations and a population of terminals, not necessarily all within range and in line-of-sight. The same quantities as listed above, measured over the terminal-to-repeater hop, will be collected using the same tools.

**Operational protocols**

**Acknowledgment protocols**

Echo acknowledgment suffers from packet interference. The delay until the echo acknowledgment is received at the transmitter is random. Thus, the packet may incur some additional transmissions beyond the first successful one creating additional overhead on the channel and in the PRUs. This number of additional transmissions is a measure of the inefficiency of echo acknowledgments; so too will be the number of packets discarded at the transmitter because of lack of reception of the echo acknowledgment. That is, the transmitter reached the maximum number of retransmissions of a packet before the echo acknowledgment was received; although the packet may have been successful, the transmitter declares itself unsuccessful in establishing communication!

Thus, we shall measure the efficiency of the Echo Acknowledgment protocol by measuring the number of additional transmissions beyond success incurred by a packet. To compute this number, a PRU must have two pieces of information; it must know how many times the packet has been transmitted, and it must also know which of those retransmissions was the one that reached the next repeater successfully. This information will be contained in two fields in each packet header, which we refer to here as fields A and B. Field B is used by the PRU to store the current transmission number of the packet. When the packet is successfully heard by the intended receiver, the contents of field B are saved by being stored into field A; when the Echo acknowledgment is successfully heard by the sending PRU, field A of the echo acknowledgment is compared with the current number of transmissions of the packet, i.e., the contents of field B in the sender's copy of the packet. If these two numbers differ, then the magnitude of that difference represents the number of times that the packet was retransmitted after it had already been successfully received at the next hop. This data is collected as part of the cumulative statistics of the sending PRU.

**Routing protocols**

Earlier we introduced the hierarchical routing scheme in use, which is based on a tree structure with the station as its root. The initial tree structure is created dynamically by the Initialization Procedure in which the station uses PRU connectivity information to create a tree that minimizes the number of hops between each repeater and the station. Thus the routing strategy initially performs shortest path (minimum hop) routing from repeaters to station and from station to repeaters. However, when the first choice shortest path cannot be used, the packet departs from this path and uses a shortest path from its new location. This will occur when a repeater has transmitted a packet over a hop the maximum number of times allowed without receiving an HBH acknowledgment; the repeater then alters the packet's header (to what is called the "ALL" label) so that any repeater
within hearing distance and able to relay the packet in its intended direction will do so. This packet is then said to be alternately routed. It is retransmitted with its "ALL" header until either an HBH acknowledgment is received or the maximum number of retransmissions is once again reached, at which time the packet is discarded.

The analysis of a routing algorithm, particularly in a broadcast, and thus mobile, network, is a complex task, in that routing is topology- and load-dependent, and involves, with varying degrees of subtlety, all of the system's protocols. Thus, routing considerations are really a synthesis of most elements of the system design, and as such, the measurement of the algorithm involves at times the study of the interaction of the many system protocols.

Given the patterns of input load on the network, the distribution of traffic flow in the net is an indication of the behavior and efficiency of the routing and initialization algorithms. One may detect the concentration of traffic on specific routes creating congestion while alternate routes are not assigned; thus smaller delay routes may have been ignored in favor of the shorter routes provided by the initialization procedure.

To obtain the distribution of traffic flow, the following quantities are to be measured:

(a) the total number of packets received and transmitted at each repeater (obtained in the PRU Cumstats)
(b) the fraction of time the transceiver is busy (obtained by snapshot statistics, or in the PRU Cumstat by regular sampling of the transceiver's state)

Also, the point-to-point nature of this routing algorithm, restricting a packet at a given hop to a single repeater as its immediate destination, does not take advantage of the broadcast nature of the channel, in which several neighbors may actually hear the transmission and be capable of relaying the packet. Thus the following quantity is relevant:

(c) the number of packets correctly received and discarded because they are destined to other components in the net (obtained in the PRU Cumstat).

Moreover, to measure the potential of each neighboring repeater (say, repeater "n") as an immediate destination, it is essential to know the probability of success $P(n)$ repeater $n$ has to correctly receive a broadcast packet. This we do by maintaining in each PRU a table counting the number of successfully received packets from each immediate neighbor. The ratio of the number of packets correctly received from a given neighbor, to the number of packets transmitted by that neighbor, is a measure of $P(n)$.

Another important feature of a routing algorithm is its adaptability to network changes: input traffic load, connectivity and component failure and repair. In evaluating the dynamics of such an algorithm, three factors must be examined: the time required to detect the network change, the time required to respond, and the quality of the response. The data items at each PRU necessary for these studies, which include some of those mentioned earlier, are:

(a) tables counting the number of packets correctly received from immediate neighbors
(b) number of packets alternately routed
(c) number of packets discarded, suggesting route congestion or component failure
(d) percent of time repeater is busy transmitting and receiving

These are obtained as cumulative statistics in the PRU.

In addition, the Pickup packet is a valuable tool in routing studies in that it contains the actual and complete route taken by the packet (pinpointing alternate routing), as well as time stamps to compute the queueing and transmission delays incurred at each repeater.

Repeater's performance

The evaluation of the performance of a repeater is most important in the analysis of network behavior; it allows us to break down key network measures (such as packet delay and throughput) into their elementary components and to examine the effects on these measures of the repeater activity and design (including buffer management, queueing discipline, and packet processing priorities).

The quantities relevant to packet delays are:

(a) The processing time of a packet flowing through a repeater; this is counted in Pickup Packets as the time lapse between the packet's arrival and the time it is placed on the transmission queue. This processing includes various checks such as checksum, packet type, routing labels, etc.
(b) The packet queueing delay at a repeater; this is also counted in Pickup Packs as the time elapsed from when the packet is placed on the transmission queue until it is considered for transmission (i.e., until it is at the head of the line, in a first-come-first-served discipline).
(c) The packet's service time; this is also counted in Pickup packets as the time elapsed from when the packet is at the head of the queue until its echo-acknowledgement is correctly received. Note that the actual service time (time until the packet is correctly received at the next repeater) is smaller than the one measured here due to the echo acknowledgment protocol used in this system. Note also that the service times of consecutive packets are correlated.

The quantities related to a repeater's communications activity are:

(d) percent of time the PRU transceiver is busy transmitting and receiving; this can be obtained in the PRU.
Cumstat by regular sampling of the transceiver's state.
(e) the total number of transmitted packets at each repeater relative to the number of successfully transmitted packets. The latter number is obtained for each neighboring repeater by examining its table count which gives the number of packets correctly received from immediate neighbors.
(f) the percent of traffic received with checksum error (obtained in the PRU Cumstats).
(g) the percent of traffic received correctly but not intended for this repeater (obtained in the PRU Cumstat).

The quantities relevant to buffer management and occupancy are:
(h) the percent of time packet buffers are in a given state (free, queued for packet transmission, reserved for packet receive). This can be obtained in the PRU Cumstat by a regular sampling of the buffer states. (i) the frequency of buffer overflow as a function of the load, and this is obtained also in the Cumstats by counting the number of packets discarded due to lack of buffer space.

Traffic characteristics

In determining the traffic characteristics, one should distinguish between external traffic (the input traffic generated by network users and traffic sources) and internal traffic (traffic relayed and generated at repeaters). The measurement functions determining the external traffic characteristics are not necessary when the entire traffic is artificially generated. They include:
(a) the geographical distribution of the input load (obtained in the end device Cumstats)
(b) characteristics of the terminal input processes (obtained in the form of histograms of packet intergeneration time from the end device Cumstats)
(c) the amount of traffic generated at repeaters for special purposes such as: control, measurement, etc. (i.e., overhead traffic) (obtained in the PRU Cumstats).

The characterization of internal traffic is crucial in the creation and validation of assumptions made in repeater models aimed at an analytic prediction of the performance of multi-repeater packet radio networks. To characterize this internal traffic, we may measure the following quantities at each repeater:
(a) interarrival time (defined as the time between the arrivals of two successive packets that have been correctly received and are destined to that repeater).
(b) interdeparture time (defined as the time elapsed between the acknowledgment of two consecutively transmitted packets).

Histograms of these quantities can be created from the information contained in the Pickup Packets.

Network's global performance

The ultimate goal of all system considerations is to create a network of high capacity providing minimal user (end-to-end) delay. We examine the success in achieving this goal by measuring the end-to-end delay and the network throughput (counted as the number of packets received at their respective destinations), under various patterns of input load, as well as the frequency of lost and duplicated packets.

It is important to note that these quantities are fundamentally affected by all the operational protocols. They allow us to obtain the main performance curves of throughput and delay.

The role of measurements in flow control

The station has the responsibility for centralized control over the entire network. To carry out this responsibility, the station requires various indications of network activity and performance. Some of this information will be acquired from incoming traffic, but much of this information must be specifically obtained by having monitoring procedures collect, from the various devices, a subset of the measurement items that have been seen presented throughout the paper.

CONCLUSION

In this paper, we have presented some of the results of our activities in the measurement aspect of the ARPA Packet Radio Project. We described the Packet Radio Network measurement facilities, consisting of the measurement tools and the techniques for data collection. We also identified and discussed the measurement functions required to gain insight into the behavior of this broadcast network. In so doing, we determined the data items required to support these functions and the means for their collection. This information is summarized in Table 1.

In the design of these measurement facilities, a constant concern is to keep the overhead they create at the components and on the broadcast channel at a low level. An important activity will be to evaluate the cost of each element of the facilities in the prototype network, and to assess their impact on the network operation so as to design and conduct experiments in a manner that will minimize the bias introduced.

ACKNOWLEDGMENTS

We would like to acknowledge the Packet Radio Working Group and particularly Ralph Jones, David Retz, Ron Kunzelman and Don Nielsen from Stanford Research Institute and Dick Sunlin from Collins Radio Corporation for the fruitful discussions that have taken place with them.
**TABLE I—Summary of Measurement Items**

<table>
<thead>
<tr>
<th>Item</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pickup Packets (at each PRU, the following data items are collected in the Pickup packet):</td>
<td>time of arrival of the packet at the PRU</td>
</tr>
<tr>
<td></td>
<td>time the Pickup packet was first placed on the transmit queue</td>
</tr>
<tr>
<td></td>
<td>time of each transmission</td>
</tr>
<tr>
<td></td>
<td>time HBH ack arrived (stored in next Pickup packet)</td>
</tr>
<tr>
<td></td>
<td>the current PRU ID</td>
</tr>
</tbody>
</table>

**PRU based Cumulative Statistics**

- # of packets received in error
- # of packets received but not intended for this PRU
- histogram of # of transmissions per successful packet
- # of unsuccessful packets (dropped because of lack of ack)
- # of packets discarded because of lack of buffer space
- # of alternately routed (“ALL”) packets received
- table counting number of correctly received packets from immediate neighbors
- # of transmissions beyond success
- # of packets incurring transmissions beyond success
- table sampling frequency of buffer states (and transceiver states)

**End-Device Cumulative Statistics**

- histogram of round-trip times
- # of packets transmitted
- # of duplicate packets detected
- # of packets discarded by the sender because of lack of ETE ack
- histogram of # of transmissions per successful (ETE) packet
- histogram of packet intergeneration time

**Note:** certain Cumstat items will distinguish between inbound (to the station) and outbound (from the station) traffic.

**REFERENCES**


Monitoring and access control of the London node of ARPANET

by ADRIAN V. STOKES, DAVID L. BATES* and PETER T. KIRSTEIN
University College London
London, England

ABSTRACT

At University College London, we have developed a novel way of monitoring a network node using a separate processor and have applied this technique to the London node of ARPANET. A monitoring program on a PDP-9 records usage of the London-TIP via the dial-up ports and these data are sent to the Rutherford Laboratory IBM 360/195 for detailed analysis. In this paper, we describe the method we have developed and present some of the results we have so far obtained.

INTRODUCTION

The ARPA computer network has been operational for over six years. During this time, there have been extensive measurements on the performance of the communications subnetwork, particularly by the Network Measurement Center at the University of California and Bolt, Beranek and Newman. There have been extensive measurements of usage of specific hosts, for example by the National Bureau of Standards. There have also been certain measurements of the network usage made for certain large applications to justify the cost of running the network. There have not, however, been any consistent measurements of network usage via one site. There are several reasons for this omission. Partly it is due to there being no mechanism by which US users of ARPANET could be forced to keep statistics of their usage, and partly it is due to there being no automatic accounting system for the use of the network.

An attempt was made in late 1974 to introduce an automatic accounting system into the subnetwork. The mechanism was that each communication computer would connect to a specific Access Control Host before it permitted a connection to be opened to any other Host. Further connections were permitted only if the correct user/password combination was given; after each session over a virtual circuit, the accounting Host was informed of the length of the connection and the number of packets transferred (unless the Access Control computer was not available, in which case the statistics were stored for later transmission). This mechanism was abandoned after a few weeks for several reasons; amongst these were the difficulty of maintaining the password file on the Access Control Host and the sluggishness of the response of the access control mechanism.

At about the same time we, at the University College London (UCL) node of ARPANET, became interested in providing access control and accounting. There were two reasons for this. Many bodies wished to analyze the extent and value of usage of the ARPANET link via the UCL node; they also wished to be able to control the access—particularly via the Public Switched Telephone Network (PSTN). Because we were a node like any other, it was not possible to put any special code into the Honeywell 316 Terminal Interface Message Processor which acts as the communication processor to the UCL site. Instead, any such code had to be provided outside the subnetwork. However, since all the use of the UCL node is purely experimental, we were entitled to enforce any extra login procedures we wished onto our users. We have been looking at three different types of measurement:

- Characteristics of usage of Hosts via the UCL node
- Characteristics of the data traffic for several specific applications
- Overheads incurred in the different levels of protocol

This paper is concerned only with the first of the above; the others will be discussed in later papers. Only sample measurements will be presented here; fuller measurements will be discussed at the Conference and are really more appropriate to a Technical Report. In order to provide meaningful statistics, we also have had to provide access controls; this subject is also considered here.

At UCL is sited a TIP; two PDP-9 computers are connected to the TIP, one of which acts as a gateway between various computer networks, in particular, between ARPA and the Rutherford Laboratory (RL) star network based on an IBM 360/195. The second PDP-9 is used both as a development machine and for monitoring and access control of the ARPANET; it also provides a simple form of access to the National Library of Medicine (NLM) Medline system. It is
the measurements made using this machine which are the subject of this paper.

OVERVIEW OF THE SYSTEM

The TIP has 8 slow (300/300 bps) and 1 faster (1200/75 bps) dial-up ports as well as three leased lines. The PDP-9 used for this work (known as PDP-9B) is connected to the TIP as a Host. It has 32K words of core (18 bit), a 256K disk and various other peripherals.

DESCRIPTION OF THE QUES PROGRAM

The program which performs the actual monitoring and access control is called QUES. This is one segment of a network system which runs on the PDP-9. When the system is initialized and has established communication with the TIP, QUES sets up a control connection to a specific port on the TIP; via this connection, QUES attempts to connect to other TIP ports (which are specified by entries in a disk file which may be modified easily). Each port may be in one of three states. The first is WILD, in which case there is no user connected; in this state, QUES may (and indeed does) make the connection. When a user dials in, this connection is broken and, on noting this, QUES remakes it and interrogates the user by asking for surname, TIP password and number of required Host (a typical scenario is shown in Figure 1). To save time, a user may give all three replies in response to the first question.

If the replies are satisfactory, QUES then breaks the connection and waits 20 seconds before attempting to remake it. If the replies are unsatisfactory, the user is allowed a second attempt then, if still incorrect, he is disconnected. It would have been possible for us to have better access control by making the connection for the user. This would require, however, a considerably heavier cpu load and to reduce this, we only make the connection in the specific case that we wish to record the whole dialogue for subsequent analysis (as we do in the case of MEDLINE; see below).

If the user gives correct answers and succeeds in connecting to the Host within the requisite time, QUES enters its third phase in which it attempts (at one second intervals) to reconnect to the port.

When the user closes his connection to the remote Host, QUES is once more able to connect to him and requests the number of the next Host required (name and password are not requested again). This procedure is then repeated continuously.

The only exception to this procedure is in the special case where the Host the user wishes to access is the National Library of Medicine (NLM). The British Library, as part of its Short Term Experimental Information Network Project (Ref. 8) has a number of centres which access the MEDLINE system on the NLM IBM 370/158. This machine is not connected to ARPA as a conventional Host, but rather simulates five interactive terminals on the National Bureau of Standards (NBS) TIP.

Since this makes the process of connection extremely inconvenient and also provides little status information, we have written a program which also runs under our network software on the PDP-9 and which automates the connection procedure. This program will be described in detail elsewhere; however, it should be noted that, if a user specifies that he wishes to access the MEDLINE system (by giving the NBS-TIP number or MEDLINE when asked for Host number), he is automatically routed to this program. If this program is unable to make the connection (or if the user states that he does not wish to use the program by specifying the Host as NLM), QUES allows the user two minutes rather than the standard twenty seconds to make the connection since the procedure is considerably more complicated in this case.

All the data supplied by the user are printed by QUES onto a paper tape (thus obviating problems such as closing files after a system crash); similarly, when the user disconnects from his Host, his connect time is recorded on the tape.

In the unlikely event that the system does crash, it is convenient to know the time, and this may prove difficult at night when there is little activity. For this reason, QUES prints a message to indicate that it is still running every half-hour.

A sample of the output from the program is shown in Figure 2.

DESCRIPTION OF THE ANALYSIS PROGRAMS

A paper tape, as in the above section, is converted into a 360 job by the addition of a few lines of Job Control Language. This job is then sent to the RL machine via our other PDP-9.

Such tapes may have many errors. For example, due to hardware problems, characters may be missing etc. Also, each interaction has generated two lines of out-

LONDON-TIP MONITORING SERVICE.
SURNAME >stokes
TIP PASSWORD >x x x
PASSWORD UNKNOWN—REENTER >indra
HOST NUMBER >42
OK—BYE
Closed * message from the TIP
@L 42 * user logs in to host
@C * close connection to host
Open * message from TIP
NEXT HOST NUMBER OR RING OFF NOW >

Figure 1—A Typical QUES Scenario
put, the first when QUES interrogated the user, the second when the user closed his connection to the remote Host.

Therefore, the first phase of analysis consists of a program called LOGB. This program checks the input for errors, removes superfluous lines (e.g., "QUES OK"), removes lines where the user has given an incorrect password or Host (these are printed out for inspection, but not passed onto the analysis programs) and compresses the data. In particular, the connect time is appended to the message generated when the user logs in. Two problems arise. The first is that monitoring may be terminated while a user is still logged in; in this case, the QUES program prints out the connect time up to the termination and so the time recorded is an underestimate. This of course presents no problems to LOGB, since there is no difference between such a message and a genuine log out. The second case, when the system crashes while users are logged in, does present a problem; in such a case, LOGB detects this by the occurrence of a "MONITORING TERMINATED" message not preceded by a "MONITORING STARTED" message. It generates the latter and the associated logout messages at the last time for which it had a valid message (hence the reason for the production of the "QUES OK" messages).

The program also produces messages indicating the number of ports in use. Due to a temporary restriction on the number of channels available in the PDP-9, QUES only monitors six ports at present. It is a simple matter for LOGB to record the number of ports in use except for one case, when QUES starts monitoring. In this case, ports may already be in use; the version of QUES to which we are referring did not attempt to distinguish whether the port was in use by a genuine user or not (e.g., someone having dialled the TIP number by mistake). This distinction can be deduced by QUES with a reasonable degree of certainty by knowing the TIP timeout period. This will be done in future, but the timeout period is not guaranteed to remain constant and the method is not completely reliable. In the data we present here this was not done, and LOGB had to deduce the number of ports in use. Due to the mode of operation of the telephone system, a user is allocated to the lowest numbered free port on dialling up. At present, we monitor ports 70 to 75 (octal).

Thus LOGB produces an output file, an example of which is given in Figure 3 (the output is that produced with the data in Figure 2 as input). This file is in a standard format and may be assumed to be error free. It is then passed to a set of analysis routines called XFSTATS for analysis. It was not expected that all the analysis functions which might at some time have been required could be specified in advance; therefore XFSTATS was written in a flexible, table-driven manner. The initial phase of the program consists of reading a file of control cards which are parsed. It then
reads the file output by LOGB, selecting only data between the dates specified by the control file. These data are mapped into a structure in core, then, depending on the control cards, various analyses are performed, for example, a matrix of connect times for each host and user (see Figure 4).

RESULTS OF ANALYSES

In a paper such as this, it is neither possible nor desirable to give a full analysis of the results we have obtained (these are available in Reference 6), and we concentrate on three aspects of the results. In the following, the data to which we refer is, in the main, that obtained in July 1975. A single month provides a reasonable volume of data covering various times of day. July was chosen specifically since it was the month in which we did most monitoring; PDP-9B is also used for system development and monitoring is only performed when it is not used for that purpose. Thus the amount of monitoring varies considerably from week to week and much of the monitoring is at night and weekends with few users; at such times, the main value of the system is for controlling improper access.

In the month concerned, we monitored the TIP for 525 hours (70 percent) broken down in the following way:

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Total (mins)</th>
<th>Mins/day (av)</th>
<th>% of time monitored</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-0800</td>
<td>9120</td>
<td>397</td>
<td>82.7</td>
</tr>
<tr>
<td>0800-1300</td>
<td>5571</td>
<td>242</td>
<td>80.7</td>
</tr>
<tr>
<td>1300-1800</td>
<td>1713</td>
<td>74</td>
<td>24.8</td>
</tr>
<tr>
<td>1800-2400</td>
<td>4426</td>
<td>192</td>
<td>53.5</td>
</tr>
<tr>
<td>Week ends</td>
<td>10674</td>
<td>1334</td>
<td>92.7</td>
</tr>
</tbody>
</table>

The three aspects with which we will concern ourselves are:

(i) The global picture: overall usage and general statistics
(ii) The pattern of usage by one specific user
(iii) The pattern of usage of one specific host

On the global picture, we monitored the TIP for 31,504 minutes in the month. The total time users were connected to various Hosts was 10,761 user minutes, giving a usage of the six ports monitored of 5.7%. However, much of the time monitored was at weekends and night and the breakdown of this figure over the time periods used above is:

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Total (mins)</th>
<th>Mins/day (av)</th>
<th>% of ports used</th>
<th>Logins</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-0800</td>
<td>8039</td>
<td>24.1</td>
<td>0.2</td>
<td>2</td>
</tr>
<tr>
<td>0800-1300</td>
<td>1443</td>
<td>4.5</td>
<td>0.2</td>
<td>3</td>
</tr>
<tr>
<td>1300-1800</td>
<td>462</td>
<td>1.7</td>
<td>0.2</td>
<td>2</td>
</tr>
<tr>
<td>1800-2400</td>
<td>723</td>
<td>1.1</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>Week ends</td>
<td>8039</td>
<td>24.1</td>
<td>0.2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1443</td>
<td>4.5</td>
<td>0.2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>462</td>
<td>1.7</td>
<td>0.2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>723</td>
<td>1.1</td>
<td>0.2</td>
<td>1</td>
</tr>
</tbody>
</table>
(where the percentage referred to is the percentage port utilization). The high figure in the 0800–1300 time is due to the non-availability of many US hosts to us in their prime shift (1300 onwards, UK time). This restriction applies to the two Hosts that were used most heavily, NLM and the Information Sciences Institute (IST) PDP-10X. A general matrix is produced of the usage at each Host by each user group. The complete matrix in our case would be 100x40 and would be quite unreadable. A partial matrix is shown in Figure 4. The two most heavily used Hosts during this period were Host 147 (64 percent) and ISI (9.4 percent). The reason that the former was used so heavily was that it had been unavailable for most of June; in addition, the general university usage was low in July because of the onset of holidays.

The pattern of usage overall is much as expected and is shown in Figure 5. In this, attempts to connect to a Host which was not available are excluded (QUES records this as a connect time of zero; there were 115 such occurrences in July). The second histogram shows the zero to ten minutes segment on a larger scale; the zero to thirty seconds time period contains many users who, although connected to the Host, were unable to log-in either due to the Host refusing to allow the log-in or rejecting it due to illegal account parameters.

It is of considerable interest to determine the number of connections made and ports occupied and these are best normalized with respect to the average number of ports in use. Typical results of the ports in use over particular periods are shown in Figure 6. We also have histogram information on the number of ports in use at any one time. This information can be used to guide the TIP owner on the number of dial-up ports he should provide.

The second aspect which we wish to examine is the pattern of one specific user (Peter Kirstein). His usage this month was confined exclusively to one Host, ISI, and there were three types of usage; these are clearly reflected in the connect times (Figure 7). The first is
usage such as reading mail or sending messages. This takes a relatively short time, of the order of ten minutes. The second usage is editing documents and this gives rise to connect times of the order of half an hour. The third usage is entering documents and, to a greater extent, teleconferencing where connect times are of the order of hours. This pattern shows up clearly in the histogram of Figure 7 but, since there were only nineteen logins recorded in the month, we also give the corresponding histogram for the four month period May-August 1975 when there were 59 logins.

The last usage we consider is that of a specific Host. As we mentioned above, a number of centres in Britain access the MEDLINE system on the NLM IBM 370/158 to perform bibliographic searches. In addition to the QUES monitoring, we are able to monitor other details since each line from the user and from NLM passes through the PDP-9. In particular, we monitor the number of characters and lines each way. We also record the number of searches carried out, but this depends on the user specifying this number accurately to NLM; also, the actual number is, to a certain extent, subjective. Therefore, we do not make significant use of this information.

In the month we are considering, we recorded 148 logins to NLM, giving a total connect time of 7385 minutes. Eliminating those logins where MEDLINE was not available (shown by a very short connect time—of the order of a few minutes—and the user specifying that no searches were done and giving no “PRINT” commands), we had 87 logins.

Although the number of observations is reasonably high, the variation in the parameters was surprising. The greatest consistency was in the time spent logged in, an average of just under 33 minutes with a rms deviation of 11 percent of this value. Two parameters which might have been expected to be fairly consistent, the average number of characters per line to and from NLM each session, showed wide variations over the recorded sessions. The averages were 16.64 and 13.65 respectively with rms deviations of 31 percent and 49 percent of those figures. Similarly high deviations were shown in the ratio of the lines from NLM to lines from the user (52 percent), the time per line from NLM (46 percent) and the time per line from the user (58 percent). These results are tabulated in Figure 8.

The results we have presented above for the use of NLM are our initial results and will be supplemented with more detail when our current extensions to the monitoring system are completed. In this, all interactions with NLM may be copied onto magnetic tape and hence a complete analysis may be performed. These data will be used partially to check the data generated by some of the users under the project of Reference 8. They will be of particular use in providing a quantitative basis for certain subjective criteria, for example, response times. It is hoped, at a later date, to extend this system so that it may be used to monitor the interactions with any Host on ARPANET.

CONCLUSIONS

The method of monitoring and access control we have developed is not of general application, particularly since it requires a dedicated processor in addition to the one being monitored and, in the general case, it is obviously simpler to put these functions in the latter. However, the introduction of monitoring and/or access controls into a computer brings with it a loss of reliability (since, not only does it increase the complexity of the software but also it requires additional hardware). This loss of reliability may not be acceptable and so our technique may be of more general applicability.

An example of this is evinced by ARPANET. Access controls were introduced into the TIPS in late 1974 and, for a number of reasons (in particular, the decrease of reliability that backing store would introduce and the problems of maintaining the password data base) it was introduced in a way that led to inconvenience for users and large network overheads. Although our approach would not obviate the problem of maintaining the data base, it would certainly not decrease the reliability of the TIPS. At the present time, the LONDON-TIP is the only node on ARPANET on which monitoring of users and access control is being carried out.

By use of this technique, we have obtained considerable monitoring data and have used this to explore the methods of usage of the network. We are currently developing other means of monitoring in conjunction with QUES to enable us to obtain a more complete view of the usage of our node. The above types of figures are useful in giving a general overview of the extent of usage of the network for different applications. The measurements give an excellent cross-check on subjective reports from user groups on their usage of specific Hosts (Reference 9). Over a period of time, we expect these measurements to be extended also to leased line ports and so to give us a complete picture of the usage of ARPANET via the UCL node.
ACKNOWLEDGMENTS

We wish to acknowledge support by the British Library under grant SI/G/093. Also, such research would not be feasible without support of the many bodies for the ARPA project at University College, in particular, the Science Research Council (B/RG/5981), the US Advanced Research Projects Agency (N00014-74-C-0240) and the UK Ministry of Defence (AT/2047/064).

We also wish to acknowledge the contributions of many people of the INDRA group at University College to this project, in particular, to Roger Gould and Peter Mott who did much of the coding and debugging of the programs mentioned.

REFERENCES


Office automation project—A research perspective

by HOWARD LEE MORGAN
University of Pennsylvania
Philadelphia, Pennsylvania

ABSTRACT

This paper attempts to place some perspective on the research and developments going on in office automation. It describes the functions which can be assisted by computers, and indicates where more research may be needed. A brief description of the Office Automation Project at the Wharton School is provided. The systems being developed include word processing, electronic mail, decision aiding technology, and integration with various databases. This effort is compared with some of the other, complementary research projects in office automation under way around the country.

INTRODUCTION

Having successfully occupied the accounting and financial centers of business for almost two decades, and the production control centers for one decade, the computer industry is finally beginning its march on the office. Until now, this was the last stronghold of human activity in many of our large business and government organizations. Office automation, of which word processing is but one element, is one of the fastest growing segments of the marketplace. A large number of companies have entered the field, and sales of word processing equipment alone should reach $7.5 billion in calendar 1976.10

This paper is an attempt to place some perspective on the research efforts going on in office automation, with some emphasis on structuring the many different services which can come under this heading. It also tries to indicate where there are problems of research rather than development interest, with the hope that this may stimulate various groups to begin work on them. Finally, it describes the efforts under way at The Wharton School in this area, the Office Automation Project (OAP).

ASSUMPTIONS

This author has based his research on the following assumptions. These are not all universally accepted, although I have found agreement on some of them in each of the major research efforts studied.

1. More efficient production of paper is not the ultimate goal of office automation. While a side benefit may be that documents, letters, etc., can be produced and changed more efficiently, one hopes to eliminate more paper than is produced in order to attain a truly automated office.

2. The burden of proof is on those making changes. Whenever we propose to make radical changes in the way in which offices function, we should be prepared to justify these changes. Technology itself is not sufficient reason to introduce change. A corollary to this assumption is that in order to have the change accepted, there must be some short term payoff to those who must use the system (managers and clerical workers).

3. Office problems are not well structured. The reason that computers have been able to take over accounting and financial departments is that the problems there are relatively well understood and structured. The type of activity and decision making which characterizes most offices does not lend itself to such clear structuring, and thus demands different solutions.

4. Both technological and organizational research challenges remain. Because of the semi-structured nature of office problems, new data structures, program structures and even hardware may have to be designed to really attack the problems. Also, since telecommunications may permit automated offices to exist in quite a different physical manner than at present (e.g., offices in the home), the impact on organizational designs and strategies require much research.

These assumptions have led us to the particular analysis and development approaches used in my contributions to the Wharton Office Automation Project.

FUNCTIONS

Top down methodology is much in vogue these days, for its supposed clarity and completeness. Here, a top
down analysis would begin with the functions performed and the people performing them. We should note that the actual system development and implementation at Wharton is being done in Ness' "middle out" approach, which tends to lead to greater user involvement and short term payoffs.

Since office functions are in some sense dependent on the particular business, we must describe the environment in which we work. The impetus for the Wharton project came with the formation of the new Department of Decision Sciences at The Wharton School. This consists of 15 teaching research faculty, about 25 support personnel (secretaries, research assistants, etc.) and has a consolidated budget of slightly under $1 million/year. There are several middle managers (typically principal investigators on projects) and a department chairman, who may be considered a more senior manager, since he must worry about the consolidated budgets from teaching and research grants. We suggest that our office is therefore similar to offices in information oriented companies, e.g., banking, insurance, R&D, rather than product oriented companies. We are aiming at aiding the middle manager in performing his or her functions. These functions fall into six major classes. For each, we discuss what has been done by the industry to automate it, and what might be done.

**Communications**

Mintzberg, in his study of managerial functions, reports that almost all of most managers' time was spent on communication. This is borne out by the fact that it is in this area that the most work on office automation has proceeded. We divide communications into a number of categories for purposes of discussion.

1. formal/informal
2. reply required/no reply required
3. message/document (short/long)
4. internal/external
5. voice/text/graphics

The word processing industry has realized the amount of time and money spent in the average office on communications and has attempted to automate some of these tasks. They have had much success with long, formal texts, as typified by legal contracts, large reports, and mass letter writing examples. The main technology involves text editors, about which we shall have more to say later.

The informal message area has proven of tremendous value to those organizations that have implemented systems such as electronic mail. In one large New York bank, for example, the time to get messages and memos from Wall Street to Midtown was cut from four hours to five minutes. The ARPANET community has also noted the powerful sense of community which the mail facility can create.

In the Wharton system, features for extracting replies are a part of the electronic mail system, but are just beginning to be used. The phone or voice communications area is untapped. Services such as logging calls, handling of phone messages, etc., could be provided. Some recent research has indicated that for office type of problem solving, voice may be more effective than face to face communications, even given the lack of graphics.

One West Coast research group is working heavily in the area of computer graphics integrated into a word processing system, and the Binary Image Processor created to store and retrieve technical manuals also is attacking this area.

**Information storage and retrieval**

Almost all of us keep files of one sort or another in our offices. Names and addresses, correspondence, task related work, all seem to accumulate in our file drawers. An informal survey of the faculty members in the Department reveals that the middle managers have on the order of 1,000 files, and the chairman has about 2,000 separate file folders. Most of us seem to be able to index the information in these files in our heads, or with simple filing schemes. The research literature on information storage and retrieval tends to lead to strategies which may be more rigid than those we use when searching our own files. For example, I may store a letter in a folder with the recipient's name on it, or in a folder with the name of a paper he or she has written, or in some other task related file. Yet I can usually retrieve all correspondence to a person by remembering the subjects of the letters.

There are basically three problems to effective storage and retrieval. First, storage of full texts is expensive. For the number of times which I typically reexamine a letter, it would be hard to justify online storage, even with the relatively cheap terabit memories. This can be alleviated by storing only the index online, perhaps with abstracts, such as is done for document collections.

Second, there is the problem of data input. Letters or documents prepared on an automated system can be easily stored. However, material received on paper from the outside incurs additional costs for input. As the use of electronic mail increases, this problem too may be overcome.

Finally, there is the problem of indexing. Automatic indexing of the type done in library systems is content oriented. The degree of success of the automatic methods depends greatly on the threshold values in the algorithms. Short letters are quite hard to index, because few of the terms occur frequently enough to exceed thresholds.

One research group has suggested that the best aid would be to simply show the user a large number of file folder headers, much as one would see them as one
opened the drawer, and let the human pattern recognition and search process be used to select from that set. Depending on the hardware, one could show from 50 to several hundred such folder names simultaneously. This scheme should work for the few thousands of files discussed so far. If we wish to integrate several people's files, this approach is not the one to follow.

Data analysis

Tools for analyzing numeric data have been available and in use by sophisticated managers for years. Regression, statistical packages, forecasting models, etc., have all been well developed to suit this market. Analysis of the other types of data which a manager sees, such as newspapers, magazines, reports, etc., is still beyond present day capabilities.

The Very Large Database (VLDB) project being sponsored by ARPA, is attempting to examine some of the problems of analyzing this “intelligence.” For example, we may wish to examine all of the data linking our company and research in computer databases. This necessitates bibliographic searches, calls to friends in other universities, etc. Learning to use even the available systems (NY Times Index, MEDLINE, etc.) is hard enough without trying to integrate these and automate access to them.

Decision making

Most middle managers are not making repetitive, routine decisions but rather are making semi-structured decisions which recur on an irregular basis. These tend to be tactical rather than strategic decisions. As part of an ONR sponsored effort we are building DAISY, a decision aiding information system. This attempts to automate the memory of an organization to support a manager making decisions, while still giving the manager the freedom to make the decision. Some examples of such semi-structured decisions are choosing an acquisition partner, deciding upon a battle plan, or trying to allocate fire fighting units in a municipality.

A budget planning process has been implemented and partly integrated with the OAP and DAISY by Purves and Godard. With this, a professor can learn how to budget for a research proposal, go through the calculations with a program, and dump the results in text format into the budget pages of a formal proposal. We are working with other decision processes, but are excited about the potential for linking this type of system to both the communications and the information storage and retrieval functions of the OAP.

Personal assistant

Goldstein has described an artificial intelligence system for aiding in preparing schedules. This type of activity is categorized as a personal assistant. It understands some portion of the manager’s decision rules, priorities, and requirements, and attempts to maintain schedules. In the Wharton OAP, Ness has implemented a simple scheduling system which permits a person to store and retrieve his or her calendar for any day or group of days. It also automates the reminder function, by printing, in priority order, items which users have placed on lists (e.g., Write letter to x, Buy wife’s birthday present, etc.).

The short term payoff and appeal of a personal assistant is quite high. It tends to draw less computer-oriented people into using the system. Other possible personal assistants might include arranging travel reservations, maintaining working paper distribution lists, and choosing appropriate referees for journal papers.

In the long run, such applications of Artificial Intelligence and knowledge based systems will be routine. They offer a great potential area for research in the fertile office automation applications area.

Linkage to corporate databases

Under communications, we discussed the interactions between people. Managers and other office personnel also routinely examine and update various corporate databases. These databases are maintained by and under the control of other offices, typically the data processing department.

It is desirable to permit the office automation system to have direct access to these databases. For example, the budget planning system described under decision making should not only feed the text processing parts of the office system, but should be able to create the proper entries in the accounting database to create the new project which has been proposed and budgeted. Other examples of such linkages might include moving name and address information from personal mailing lists into corporate customer databases, and examining personnel databases for skills required on particular projects.

The key research problem posed by providing this function is that of designing a simple query language for the databases. This is being examined at the present time by the End User Facilities Task Group of the CODASYL Committee.

PRODUCTS

The functions described above have led to the development of a number of products in attempts to automate one or more of these. First and foremost are text editors. These have been around for a long time and are the backbone of most timesharing systems. Unfortunately, few of these line oriented editors are very good for office automation. Several major research
groups (SRI-ARC, Xerox, USC-ISI) have all realized that good editors require separation of the editing commands from the text being edited. This can be done on CRTs, but not on the typewriter-like devices which most timesharing systems have to support. The NLS system is an excellent example of sophisticated editing. This system also changes the way in which people think about documents, emphasizing the hierarchical nature of long documents, and the ability to view a document at various levels of detail, i.e., chapter headings, section headings, etc., down to sentences and paragraphs.

The sophisticated editors at another installation not only separate command and text being edited, but also always try to show the user final formatted copy, even if this involves multiple fonts. The VyDec word processing system also works in this mode, which is quite appealing to secretaries and other initial document entry personnel. Since they can see an attractive, finished form as they enter a draft, a reward is present, reinforcing their desire to use the system.

One conclusion that can be drawn from studying the best editors available is that they require much more in the way of CPU and memory resources than most people estimate. To present an 8.5 by 11 inch screen with several type fonts may require one million bits of storage for the image alone.

Electronic mail packages are provided on ARPANET and have been developed in house by a number of companies and timesharing bureaus. Except for the Wharton system, most of these seem to treat all mail the way the US Postal Service does, i.e., as first class mail, without any real indication of the type or content.

Integrated word processors are machines combining an editor with some minicomputer and storage devices. These go under a number of names, but are really automated typewriters. Unfortunately, most of them lack communications ability, and hence could not serve for the full range of functions mentioned in the FUNCTIONS section.

Database managers for handling the semi-structured types of information used in organizations are available. Many firms market name and address list maintenance programs, and the general database management systems can easily be used for these purposes.

It is important that these systems permit easy addition of fields, storing of textual information, and extraction from files.

These are just some of the products under development or available for commercial use. We next describe briefly the first product of the Wharton OAP, the Office Automation System (OASYS).

OASYS

Figure 1 is a block diagram of the major functional pieces of the Wharton Office Automation System, designed and developed for the PDP-10 in MACRO by Professor David Ness. Those marked with asterisks are being reprogrammed or added by the author and others, using higher level languages, and DBTG database technology.

The text processing and runoff systems are a set of extensions to the DEC Runoff program for producing formatted text. Some of the more interesting features created by Ness are in the personalization, history and profile areas. Each of the programs in OASYS is designed to deal with several people. The "for" user is the manager for whom the program is being used. The "by" user is the person who is actually at the terminal, e.g., a use of the schedule program by a clerical worker for a manager. The profiles tell these programs what input forms the by user desires depending on level of skill, etc., and what output forms the for user demands. The history features permit the system to inform each user when any of the questions or commands have changed, or when new features are added to the system. This is done by recording for each user, module, and command, a level number. Before the system asks a question, it determines whether or not the user has seen this version. If not, it is prefixed by (NEW).

The profile permits users to further personalize the modules by specifying a user/module combination which questions or prompts may be skipped. Thus, in creating name and address files I am asked for home address, which I routinely keep, but Professor Ness is only asked for business address.

The electronic mail system has several types of mail. A user is told how much of each class is waiting for him or her at login time. Currently, the system permits us to distinguish among BUG mail, relating to system bugs, regular mail, return receipt mail, which auto-
TECHNOLOGIES

One of the key questions in office automation is the type of hardware and software technology which should be used for developing products. Wharton and SRI-ARC have both chosen to work on large central processors with communications facilities and relatively unintelligent terminals. Others have chosen more distributed technology, emphasizing minicomputers and some communications ability. This author feels that the latter route is the one likely to triumph in the medium term, but that we are likely to see a relatively large minicomputer supporting a cluster of terminals in one office, rather than one at each terminal. All of these minicomputers would, of course, be connected together for mail and database access.

As noted under products, some form of video display terminal (CRT, plasma panel, etc.) which can handle full graphics is required for the long term solution. It is not clear whether or not we should retain the present aspect ratio dictated mostly by the availability of cheap television (525 line scan) devices. In any case, the speed with which the display communicates with the machine for output should be as high as possible. Graphical input devices are also required, and the "mouse" - a device which is rolled around on a hard surface and tracked by a cursor on the display, seems to be an attractive current answer.

On the software side, systems have been developed in languages ranging from assembly to INTERLISP. One would hope that higher level languages would be used more extensively, but since editing is often character oriented, reasonable efficiency can be achieved on many machines only by staying close to assembly level. Languages such as BCPL and BLISS permit this, while still retaining advantages of high level languages.

Sorting is such a fundamental operation that it must be considered a part of the software technology. In a similar vein, good man/machine dialogues are needed for any system which is going to have unsophisticated users. Martin\textsuperscript{19} describes these techniques in quite a bit of detail.

RESEARCH AREAS

While much has and continues to be done in the office automation area, I have identified three major technological and two major organizational areas which are ripe for research. Some of these areas are beginning to be examined by members of the OAP at Wharton.

1. Input processing. Almost all of the functions which have been automated involve the production of output. How can we efficiently handle input in automated ways? Some OCR technology, some automatic indexing technology, and man/machine interaction technology must be brought to bear on this problem. Even if we begin to receive a large portion of our mail electronically (and several ARPANET users now get more than 50\% of their mail through the network), the time and difficulty of entering externally supplied information must be cut down.

2. Integration of databases. Integrating databases more complex than names and addresses is not available in most office automation systems. How to present a clear and simple interface to the end user is the major problem here.

3. Using artificial intelligence methods. We have begun work on a knowledge based system to assist me in processing papers for the journals on whose editorial boards I serve. This will "understand" the types of interactions I have with authors, referees, and other editors, and will try to generate the proper correspondence, database updates (i.e., correspondence logging), and action requests. There are many other places where AI can contribute.

4. Amount of personal communications needed. Organizational studies on the effectiveness of dispersing the office personnel to separate locations, communicating only through the system, must be made. Chapanis\textsuperscript{2} has done some work on problem solving with various communication modes, but we really know very little about the best ways to do this. Should the people meet face to face once a week? Once a month?

5. Who gets the terminals. For years people in the MIS field have seesawed on whether or not the managers or their staff assistants will actually have the terminals on their desks. Someone should attempt to answer this question through controlled experiments.
SUMMARY

Office automation is a growing area of concern for computer and organizational researchers. The efforts to date have focused mainly on word processing, which is only a part of one of the office functions. Much more attention needs to be directed to the other, higher impact, less structured activities which take place in offices. This should yield a set of high quality products which will change the way in which offices are run over the next two decades.

The Wharton Office Automation Project is attempting to attack some of the non-word processing areas, paying particular attention to database integration, good user interactions, and handling of decision processes. We realize that our comparative advantage is not in the hardware area, and hence are trying to build modules of a system which can be grafted onto good hardware as it develops.

It is only a matter of time before most of us will be interacting with other people with the aid of computers as an everyday occurrence. We owe it to ourselves and the field to ensure that this happens in an efficient and socially responsible manner. This paper has not discussed the obvious privacy/security requirements which will be necessary in office automation systems, but they must be thought of at the beginning of any design projects.

This paper has attempted to set down something of the state of research in the area in late 1975. Functions, products, and technologies have been characterized with an eye towards possible research topics. The author hopes that those who work on these topics will communicate with him.

ACKNOWLEDGMENTS

The Office Automation Project was the result of necessity and a number of discussions among Professors David Ness, E. Gerald Hurst, Jr., Thomas Johnson, Rob Gerritsen, and myself. Dave Ness has spearheaded the effort and did almost all of the programming for the first generation OASYS. The author has benefited from the above individuals' comments, as well as the recent programming and intellectual support of O. Peter Buneman and Michael Zisman, and the editorial assistance of Dorna Caskie.

REFERENCES


Evaluating the impact of office automation on top management communication

by JAMES H. CARLISLE
University of Southern California
Los Angeles, California

ABSTRACT
This paper is concerned with the impact of new communication technologies on the effectiveness of top management decisionmakers. Word processing technology is only the beginning of a revolution in office automation and managerial communication which will include teleconferencing, electronic mail, and wide availability of personal computer-based systems. The potential problems and benefits must be considered in the context of the overall communication system and management needs of an organization. A research methodology is described which leads to the development of organizational models within which: (1) management communication problems can be anticipated, (2) solutions can be proposed and compared, (3) specific office automation systems can be designed, and (4) the impact of alternative systems on organizational effectiveness can be predicted and evaluated.

INTRODUCTION
Office automation is one of the new buzzwords used to describe the computer augmentation of day-to-day office functions. Most of these functions involve some aspect of the increasing volume of interpersonal and formal communications which take place within organizations. Facsimile, electronic mail, word processing, teleconferencing, on-line calendars, information storage and retrieval and general management information systems are available today to top management and even to middle management in many organizations. The combination of these (and more) functions into an integrated, computer-based system for use in managing organizations is likely to bring about an organizational revolution for white collar workers comparable in magnitude to that resulting from the introduction of the assembly line to blue collar work.

Anticipation of the organizational impact of this new communication technology is essential to the long range planning of both system developers and eventual user organizations. There is currently a lack of any generally accepted theory and methodology for evaluating the impact of today’s office automation systems. Between now and the time when computer terminals and digital voice coder boxes sit on every manager’s desk, there is an opportunity to study and improve the effectiveness of such systems. We need to study how and why the early systems are being used. More importantly, we need to understand how these systems impact the overall communication system of an organization and its management.

This paper describes the design of a scientific analysis of top management communication. This analysis results in a model of communication activity for the particular organization and manager studied. This model then supports a strategic analysis of management problems arising out of increased demands for communication and the evaluation of new communication technologies coming onto the market to “solve” those problems.

The orientation of the research described in this paper is top management effectiveness, not computer technology. Office automation should not be studied as an end in itself, but as a means to improved managerial performance. Top management stands to gain or lose the most from this new technology. Their valuable time and energy is largely devoted to communication activity of one sort or another. Freeing up time for top management and giving them increased potential for effective and rapid communication within and among organizations are attractive payoffs. On the other hand, if office automation is developed and introduced incorrectly, it could disrupt office communications, making them even less efficient, and create serious resistance to all of the new communication technologies.

If office automation fails to gain acceptance and demonstrate value in the executive suite, it is unlikely to receive adequate financial support to be properly used throughout the organization and throughout society.

THE NEED FOR OFFICE AUTOMATION
One of the most significant problems facing management in the coming years is the rapidly increasing
complexity of the information environment in which both long range and day-to-day decisions must be made. In his article on “The Future-Oriented Corporation,” Dr. Burt Nanus observed that

We live in an age of increasing complexities, an age of macrosystems in which everything of importance that occurs anywhere in the world is immediately known everywhere else, thereby precipitating consequences which, in their own turn, provoke still other changes. The recent energy crisis is an excellent example of the interrelationship of political, economic, and technological factors and the enormous significance for corporate decision making.¹

So far these crises have been few and temporary in nature. Structured management information systems have, unfortunately, proved inadequate to supply the information needed. Much of the communication in such crises is unstructured, informal, and contains subjective and up-to-the-minute data. Today’s organizations, with their traditional methods of processing letters and reports and their reliance on telephone, telex, mail and personal meetings for informal communications are ill-equipped for the volume of critical information flow during even a minor crisis. The problem is already with us, since past crises have been merely amplifications of the normal day-to-day flow of structured and unstructured communication within our increasingly interconnected communications environment.

Few managers escape the daily avalanche of unstructured communications from both inside and outside their organization. Fewer still have a clear enough model of their communication environment to support efficient and effective utilization of their time through scheduling and monitoring selectively those communications with the highest priority. There are secretaries and managerial assistants who perform this function quite well. The drawbacks to this solution are that it is becoming increasingly expensive and it leaves the manager highly dependent on that other person.

POTENTIAL PROBLEMS IN IMPLEMENTATION OF OFFICE AUTOMATION

The realization that we are rapidly entering an era in which information and communications are the limiting factors in performance of a majority of organizations is reflected in the recent excitement over office automation. Corporations are reaching for the elusive carrot of “cost reductions in word processing”—often irrationally and often with little understanding of the far reaching effects that the new communication technologies are having on their users. Most automated office projects up to this time have been fiascos.² Major suppliers of the hardware and software for “word processing” seem to know almost nothing about the informal organizational environment into which their systems have brought chaos. Not only have customers had serious organizational problems, but even the vendors themselves have had trouble with inhouse implementations of their text-editing and document preparation systems. The computer industry vendors seem to think that management communications can be improved just by speeding up the production of error-free text.²¹ It’s important to keep in mind that the “word processing” aspect of office automation is to communication automation what keypunching was to management information systems—only the tip of the iceberg.

There is a strong possibility that all the mistakes of the management information system era of the ’60s will be repeated with office automation due to the zealous marketing of computer companies and management consultant firms hoping to jump on the bandwagon of this new communication technology.

One of the reasons which has been most frequently pointed out for the failure of management information systems and management science in general has been the lack of understanding on the part of top management itself of the theories on which these decision support capabilities were based. A second reason is the lack of understanding on the part of the system developers of how people within the organization actually do their work. Too many computer-based systems have already been designed on the basis of technological breakthroughs and innovations which were insensitive to the limit on man’s rationality and the social needs that must be satisfied within organizational structures.

It is as if Russell Ackoff’s plea to avoid the development of “Management Misinformation Systems” was ignored by computer system designers.² It is as if Chris Argyris’ analysis of “Management Information Systems: The Challenge to Rationality and Emotionality,” was understood only by those managers who had personally suffered the neglect or outright sabotage of management databases.²⁶ The computer industry has not responded adequately to the real challenges of providing useable and responsive management decision support systems. We are now at a point in time where,
fastest growing industry in the United States, with an average annual growth rate of over 10 percent. Recent introduction of digital data networks such as the ARPA network, and value added networks such as TELENET, offer new telecommunication services which are highly competitive with telex, telephone, and the post office, as well as a viable alternative to at least some business transportation.

WHY IS RESEARCH ON MANAGEMENT COMMUNICATION NECESSARY?

The question of paramount importance is how do we get to the office of the future without destroying the social fabric of today's organization and without further dehumanizing communication within our society? To answer this question we must have a better understanding of the office of today, based on scientific examination of management behavior. The needs for any new communication technology should then be derived from analysis of that behavior and consideration of personal and organizational values.

Most of what little we do know of managerial behavior has been learned with primary concern for leadership style, rationality of decision-making and determinants of satisfaction and motivation. The communication needs and behavior of management are only a recent focus of investigation. Henry Mintzberg, who conducted one of the most extensive investigations of managerial work, recently noted that

I was struck during my study by the fact that the executives I was observing—all very competent by any standard—are fundamentally indistinguishable from their counterparts of a hundred years ago (or a thousand years ago, for that matter.) The information they need differs, but they seek it in the same way—by word of mouth. Their decisions concern modern technology, but the procedures they use to make them are the same as the procedures of the nineteenth-century manager. Even the computer, so important for the specialized work of the organization, has apparently had no influence on the work procedures of general managers. In fact, the manager is in a kind of loop, with increasingly heavy work pressures but no aid forthcoming from management science.19

With the exception of Mintzberg's study, which proposed and supported an intriguing theory of managerial work roles, most of the research on management communication has attempted to measure attitudes and preferences among alternative communication media. This latter research has been supported or conducted primarily by the telephone companies of the U.S., Canada and Great Britain as a form of market analysis for picturephones. It should be noted that attitudes are not always correlated with behavior. Even behavioral studies with college sophomores conducted in research laboratories offer only limited insight into the needs and behavior of top management in today's organizations, let alone the office of the future. We currently lack theories and models with substantial empirical support with which to anticipate and evaluate the impact of office automation on organizational communication systems.

WHAT ARE THE OBJECTIVES OF CURRENT RESEARCH?

The remainder of this paper describes a project under way to anticipate and evaluate the impact of office automation on top management communication in a large decentralized organization. This project is designed to lay the groundwork for management planning concerning the use of information technology to support unstructured management decision making and communication. The primary objective of the project is the development of a behavioral science model within which

- management communication problems can be anticipated,
- solutions can be proposed and compared,
- specific office automation systems can be designed, and the
- impact of alternative systems on organizational effectiveness can be predicted and evaluated.

Such a model provides top management with an alternative to computer vendor systems analysis. This project incorporates several theories of organizational behavior and human performance. It is directly responsive to recent proposals for designing more people-oriented computer systems. Most importantly, the model addresses the complexity of management behavior and of the organizational environment onto which any computer based management support system might be imposed.

This approach to the improvement of managerial communications is based on the contentions that (1) human resources in organizational management are more valuable and less well understood than the hardware and software that might be designed to support them, and (2) that person-computer communication can best be understood and improved by developing a better understanding of how people communicate with each other.

This is an ambitious project. No behavioral science or management science theory has offered a viable solution to the problems we are addressing. Surprisingly few have even tried. Our confidence in success is based on development of a new research methodology and on asking, at the outset, what we believe to be the right questions. For example, the models developed in this project should be able to answer the following ques-
tions regarding the need for or evaluation of alternative management communication or decision support systems:

1. How would the nature of management behavior be changed?
   A. How would overall efficiency be improved?
   B. How would overall effectiveness be improved?
   C. How would tasks be redefined by the manager?
   D. How would behavioral alternatives be redefined and expanded or limited in number?
2. How would manager interaction be changed?
   A. Which responsibilities and interdependencies would be affected?
   B. Which communication patterns would be affected?
3. How would resource consumption be affected?
   A. How would travel patterns be altered?
   B. How would telecommunication patterns be altered?
   C. How would computer resources be utilized?
   D. How would secretarial resources be utilized?
   E. How would managers' time allocation be affected?
4. What would the impact be on attitudes and morale in the organization?
   A. What aspects of the system would meet with strong resistance?
   B. What aspects of the system would be readily accepted?
   C. In what ways would the system support managerial growth in communication?
5. How would the important relationships between management support, management behavior, and managerial performance be affected?

The development of models to guide the examination of these and related questions is a necessary first step to rational strategic planning for the office of the future. As Peter Drucker has pointed out,

_The future manager will find the computer as much a fact of life as children today find the telephone . . . the computer is a tool of liberation if used correctly. Otherwise, you become its servant. It should liberate you from being chained to operations and to your desk and enable you to have time for people and for the outside, where the results are._

Drucker's optimism is encouraging, but unfortunately not supported by the history of the computer industry. Improperly anticipated and poorly designed, the office of the future may lead to alienation, job fragmentation, regimentation and the 1984 horror of monitoring of all electronic communications. Properly designed, the office of the future could, instead, increase the manager's control over his or her information space, expanding the rich array of communication channels and formats available for effective organizational management. This project is investigating unstructured managerial communication as a first step in anticipating and evaluating the impact of office automation on organizational communication systems. Four basic goals guide this research:

1. Increased ability to deal with more complex information environments without increasing the number of managers in an organization.
2. Freedom for the manager from his or her desk and office as the central communication and information processing station.
3. Consolidation of management communication, scheduling, and decision making activities and support technology.
4. Increased effectiveness and efficiency in dealing with unstructured management communication tasks.

**RESEARCH METHODOLOGY**

The methodology used in this research is both empirical and theory-driven. It involves unobtrusive observation and analysis of managerial behavior, on the job, over a period of one or more weeks. From this analysis, a scenario is constructed which highlights key management communication activities. Both structure and content of these communications are analyzed to identify opportunities for increasing managerial effectiveness and efficiency. The critical part of the analysis is the construction of models of the individual manager's task and communication structure. Based on these models, additional scenarios are constructed, showing the impact of alternative management communication support technologies for each manager. This approach is holistic in that the full range of organizational and decision making activities is considered. It uses case study analysis to model the complex reality of individual managers.

This data collection process, referred to as "structured observation," derives from anthropological and sociological research. More recently a research group at the USC/Information Sciences Institute has been using a form of structured observation to study human dialogue as a means of improving man-machine interaction. The most significant application of the structured observation methodology in the area of management work activity was a study conducted by Henry Mintzberg, while a graduate student at M.I.T. That study involved the detailed observation and analysis of work activity of five chief executives over a period of a week each. His results provide important motivation and direction for the development of models of managerial communication. Whereas Mintzberg decided to omit from his analyses all interaction between the manager and his secretary and to classify individuals with
whom the manager interacted only as outsiders, superiors or subordinates, this project will analyze communications with respect to specific individuals and tasks.

The theoretical foundation on which data collection and analysis are based in the current project includes a novel conceptualization of managerial activity as communication acts intended to accomplish some specific purpose. These purposes are characterized as tasks which may be in various stages of completion, once initiated. The actual model of tasks, their states, and the effects of individual communications will be formulated out of the analysis of structured observation of individual managers on the job. Thus, the data collection is partially structured prior to its initiation and is augmented by interpretations made during and after the actual observation.

One of the compelling virtues of this methodology is its inclusion of far more of the manager's work environment than a typical laboratory study or questionnaire survey would permit. This approach requires cooperation and a high degree of trust between researcher and manager, but offers significant joint learning opportunities for both. This methodology requires a minimum of conscious effort on the manager's part to generate data while maximizing the opportunity for interpretation of ambiguous behavior by the manager. Perhaps most importantly, the methodology eliminates any need for deception by the researcher and assures the manager of full confidentiality at all stages of data collection and analysis. The most serious drawback of the methodology is the enormous amount of time involved in coding and recoding detailed observation data. This extensive analysis is justified by the insight gained into the managerial process with respect to the particular questions and problems being investigated.

Following preliminary analysis of the structured observation data, models of the manager's communication activity and sample scenarios are presented first to the manager and then to a group of managers in the organization. These scenarios are revised and personalized by the group as part of the generation and evaluation of ways for improving unstructured top management communication in the organization. There is considerable evidence that managers react to idealized scenarios about the future with considerably more insight and enthusiasm than they respond to other needs analysis techniques (such as questionnaires and interviews) or functional specifications of proposed systems. 15

The collection of scenarios agreed upon as representing information processing activities for a variety of managers are then used to define a set of primitive information processing capabilities and a set of necessary hardware capabilities for making the scenarios possible. These primitives and this functional analysis form the basis for an information system design to support and improve managerial communications.

CONCLUSION

An important advantage of this methodology is that it gets many, if not all, of the organizations' top management involved in the iterative design of their own communications system. When an operational system is finally available they should already be knowledgeable as to its functional capabilities and cognizant of the likely organizational impacts.

The action research approach described above leads to the development of systems custom-tailored to the needs of users, even if the primary hardware and software still come from established computer manufacturers. Several key characteristics of the approach deserve reiteration:

1. The spirit of the project is one of cautious and rational planning for the future by development of a necessary understanding of today's top management communication activities.
2. System design proceeds by interpretation of an holistic model of managerial activities. Scenario evaluation and development brings out the values and perceived needs of the managers themselves.
3. Structured observation of management communication behavior and preliminary design specification are not "technology driven," but reflect characteristics of the client organization.
4. The research methodology has the open intention of gradually educating the managers themselves so that they can contribute knowledgeably and efficiently to any eventual system design and implementation.
5. The project focuses on top organization management on the assumption that their time is most valuable to the organization and thus can most significantly be affected by the quality of unstructured communications support systems.

REFERENCES


The evolving market for word processing and typesetting systems

by J. CHRISTOPHER BURNS
Arthur D. Little, Inc.
Cambridge, Massachusetts

ABSTRACT

The word processing, text editing and typesetting industries have become an important market for computers and computer software. Industry installations are described and sales are forecast for major systems components over the next five years. An evolution is suggested which will link word processing, in-plant publishing, text editing and business data processing systems over the near future.

More than 100 billion words are set in type each year in the United States—about 10,000 times what an average person could read if he did nothing all year but read. There are two interesting facets to this figure: first, it seems to be rising, not only in absolute terms but in proportion to the population. As nearly as we can tell words set per capita in the United States has risen 16 percent in the last ten years, this in spite of increased television broadcasts and a decreasing percentage of the population attending public and private schools and colleges.

The second facet is that with all this information to exchange we are still choosing to set it in type. Over the past ten years we have seen the development of inexpensive display terminals, high-speed non-impact printers and microprocessors which could bring this information directly into the home, bypassing the centuries-old tradition of typesetting. Yet, except for specialized financial applications, we are not using the new technology. Today nearly 70 percent of what you read has passed through a computer in machine readable form at least once, and yet the product does not differ much from its 14th century Chinese ancestor. We are here talking about typesetting not as a dusty curio but as a market for computers, and the reason for this is fundamental to an understanding of how the market will evolve and the demands it will make on the successful vendor.

Let’s look for a moment at typesetting: This is an example of how the news might look if it came over communications lines to a home terminal (Figure 1) and this is how the news looks on a typical newspaper page (Figure 2).

Or this example: An entry from a parts catalog as it might appear on a computer printout (Figure 3) and the same data displayed using typography (Figure 4).

The point, of course, is that typesetting makes it possible to mix sizes, type styles and different layouts in order to present information with greater efficiency, leading the eye, providing ready visual tags to aid retrieval, defining the nature of the information and organizing it for ready reference. Typography is a complex language which can support or overwhelm the message it carries, and mastering this language requires similarly complex commands.

The catalog entry shown, for example, required 104 separate instructions imbedded in the text to select type style, size, line length and so forth, all in composing a block of about 210 words. The development of computer systems to control typesetting has been a surprisingly difficult task. In the ten years since the first work was done on computer assisted typesetting we have only accomplished half the things we knew then to be possible. Hyphenation and justification have been done as well as stored formats, tabular work, run arounds and rudimentary pagination, but there is still no generally accepted system to handle the simultaneous composition of multiple columns, copy fitting, layout assistance, complex chemical and technical typesetting or proofreading, through each was foreseeable as early as 1967.

Typesetting systems today are about where business data processing systems were in the days of the 1401, batch oriented, close operator involvement, lots of home-grown software around, little or no full systems integration and occasionally brilliant installations in a general population of ill-fitted, commonplace and troubled efforts.

There are about 800 such typesetting systems installed in the United States today (Figure 5). The earliest of these—the IBM 1130—was equipped with excellent field developed software and became widely popular in the late 1960's although it was only a pro-
Arthur D. Little, Inc. is one of the world's oldest, largest, and most diversified research, engineering, and management consulting organizations. Established in 1880, ADL has a staff of more than 1,500 personnel with headquarters offices and laboratories in Cambridge, Massachusetts, offices in Washington and San Francisco, and subsidiaries in Canada, Brazil, London, Paris, and Brussels. More than half of our staff is comprised of scientists and engineers representing nearly every field of science, technology, and business; we are thus able to provide a uniquely comprehensive approach to the solution of scientific, technological, management, and economic problems./\n
We have available within our organization a combination of highly qualified technical, scientific, and managerial talent; extensive laboratories, computer facilities, and fabrication facilities which permit us to carry a project from the initial feasibility and system analysis stages through to the fabrication and testing of experimental prototypes or completed special-purpose hardware./\n
We are, by the nature of our organization, committed to the purely objective treatment of those problems we undertake to investigate. As an independent, profit-making research organization, we are critically dependent upon originality of thought and impartiality of judgment in prosecuting our work. Our task has often been to define
Power Pioneers
Some Small Innovators
Heat Homes by Sun,
Light Them by Wind
Experimenters Find Gadgets
Expensive but Satisfying;
Research Funding Elusive
Disguising a Tank as an Urn

By DAVID BRAND
Staff Reporter of THE WALL STREET JOURNAL
Tijeras, N.M. — So far as Robert Reines is concerned, Arab countries can turn off the oil tomorrow and electricity costs can go through the roof. He is insulated from all that.

Mr. Reines lives in a igloo-like white dome that gets its heat from the sun and its electricity from the wind.

The dome home is on a hillside some 20 miles from Albuquerque. At night, with the inside lights shining through the dome’s portholes, the scene is reminiscent of an H. G. Wells futuristic novel. In fact, Mr. Reines sees himself as pioneering an age when whole communities will be energy-sufficient.

“I have freedom because I have all the energy I need, and that’s real freedom,” he says with a fervor that almost brittlely.

Sun Businesses Blossom
Already in the U.S., according to a University of Colorado survey, there are nearly 200 solar-heated houses built, under construction or planned. To equip these homes, more than 100 solar-equipment makers have emerged, many of them small, backyard concerns. The solar-power market, according to a study by the research firm of Arthur D. Little, could reach $1.3 billion by 1985. If industry, with effective government support, moves ahead promptly to introduce solar hardware into the marketplace.”

Now this hardware is high-priced. It’s largely handmade from expensive materials such as copper, and large amounts of the materials are necessary. A commonly used rooftop solar panel is a glass-covered aluminum or steel tray in which antifreeze, water or air is heated as it moves through blackened copper tubing. In many parts of the U.S., such heat collectors must cover at least 50% of the roof surface to provide about 80% of a house’s central heating.

Bonanza forecast for US communications

CAMBRIDGE, MASS. An explosive growth in electronic business communications, brought about by a combination of advances in technology and changes in regulations, has been forecast by two members of the staff of the Research Funding Elusive. 

ADL Predicts U.S. Telecommunications User Bonanza

“In the early 1980s, U.S. telecommunications users in business and government will be treated to a bonanza—new services, more service options and flexibility, much wider choice among suppliers and significantly revamped rate structures. In some cases they will literally be able to choose among suppliers and significantly revamped rate structures. In some cases they will literally be able to order services by business and government were approximately $18-20 billion, compared with mailing costs of $8-9 billion and travel expenses of $8-9 billion and travel expenses of $16 billion. Even though telecommunications expenditures may even slow in 1975, they should experience healthy growth through 1980, reaching some $30 billion by then.

User Bonanza

According to Roetter and Shapiro: “In 1974 total U.S. telecommunications expenditures by business and government were approximately $18-20 billion, compared with mailing costs of $8-9 billion and travel expenses of $16 billion. Even though telecommunications expenditures may even slow in 1975, they should experience healthy growth through 1980, reaching some $30 billion by then.}
DUAL-CAPACITY GAS WALL FURNACE
WITH HI AND LO HEAT; 2-SPEED FAN $229.95
50,000 BTUH

5) TWO-SPEED FAN CYCLES AUTOMATICALLY BETWEEN HI AND LO FOR HEATING. CAN ALSO BE USED FOR CONTINUOUS AIR CIRCULATION IN SUMMER WITHOUT HEATING. TWO-STAGE BURNER IGNITION. MANUAL SELECTOR ON GAS VALVE FOR HI OR LO GAS INPUT.

BEIGE ENAMELED STEEL CABINET. VINYL WOOD-GRAINED CONTROL PANEL. CAN BE WALL-MOUNTED OR RECESSED INTO SINGLE STUD SPACE. VENTS FROM DRAFT DIVERTER AT TOP. USE 4-INCH GAS VENT KIT 42 AY 7367N (SOLD BELOW). ORDER OPTIONAL REAR REGISTER KIT AND BACK VENT KIT BELOW. FURNACE MEASURES 72 INCHES HIGH, 14 INCHES WIDE.

50,000 BTUH MODEL: 10 1/2 IN. DEEP WALL-MOUNTED; 6 5/8 IN. DEEP RECESSED.
FAN 300 CFM MAXIMUM USES 106 WATTS; 65,000 BTUH MODEL: 14 IN. DEEP WALL-MOUNTED; 10 1/8 IN. DEEP RECESSED. FAN 435 CFM MAXIMUM USES 186 WATTS.

High Input Low Input Natural Gas LP (BOTTLED) Gas Shpg. Wt. Price
50,000 BTUH 30,000 BTUH 42 AY 73631N 42 AY 73636N 100 POUNDS $229.95
65,000 BTUH 40,000 BTUH 42 AY 73632N 42 AY 73637N 125 POUNDS $264.95

REAR REGISTER KIT FOR FURNACE (5) ABOVE. SUPPLIES LIMITED HEAT TO ROOM BEHIND FURNACE THROUGH REAR WALL. INCLUDES REGISTER WITH DAMPER AND FITTINGS.

42 AY 7310 - SHIPPING WEIGHT 5 POUNDS.......................... KIT $12.00

Figure 3

Figure 4

In Group D are the larger systems in the market today: DEC's Typeset 11, the large Hendrix text editing system, SDC and the large Tal-Star system. Products in this group typically support 32 to 64 video terminals and have been sold almost exclusively to large metropolitan newspapers. Some of them do limited business data processing.

The last group includes the very big systems. DEC has written a full typesetting system for its PDP-10. Harris has built several large systems based on the PDP 11, and there are several major daily newspapers who have written their own software for the IBM 370.

A note on systems which do not appear on this table: IBM has announced but has not delivered a typesetting system for the 370 called Printext 370. Univac has announced a typesetting system called Newscom. Dolphin Graphics has acquired the U.S. marketing rights for MOPAS, a European system of some promise and there are at least six other suppliers of typesetting systems in varying stages of specula-
report that conversion from hot metal to computerized photocomposition has cut as much as 50 percent off their composition and production costs, often equivalent to an increase of 10 percent in profit before taxes.

Over the next five years the market is likely to behave somewhat differently. A major increase in small word processing and text editing systems sales is expected, with floppy disk-oriented systems available for under $50,000. Optical scanners are likely to lose sales momentum and may in fact experience declining sales by 1980, functionally replaced by communicating terminals and word processing equipment. And I think it’s clear now that Xerox, Redactron, IBM, Digital Equipment and others intend to turn memory and display typewriters into a billion dollar a year business by 1980 with the probable result that at least 25,000 communicating and display typewriters will become terminals for typesetting systems. Phototypesetters will continue to sell, though at a lower price. At the center of this growth is an enormous potential for small and distributed computer systems which not only can meet the composition needs of the user but also can perform file management, message switching, data storage and retrieval by key word or subject as well as inter-computer communications.

It is probably useful at this juncture to point out that there are more than 30 suppliers now trying to capture a share of this market, with only four achieving annual sales over $10 million and at least 12 operating on sales of $2 million or less.

Who will the customer be over the next five years? (Figure 7) Certainly the printing and publishing industry is the primary target and within that the daily newspaper continues to be the most attractive sector, although our own figures are now suggesting a greater saturation of that market than most other forecasts indicate.

<table>
<thead>
<tr>
<th>Market Sector</th>
<th>Number of Establishments</th>
<th>Annual Composition Volume</th>
<th>Percent of Market</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily Newspapers</td>
<td>1,770</td>
<td>970</td>
<td>55M Words</td>
</tr>
<tr>
<td>Weekly Newspapers</td>
<td>9,500</td>
<td>300</td>
<td>.9M</td>
</tr>
<tr>
<td>Magazine Publishers</td>
<td>2,500</td>
<td>500</td>
<td>.5</td>
</tr>
<tr>
<td>Book Publishers</td>
<td>1,200</td>
<td>300</td>
<td>1.9</td>
</tr>
<tr>
<td>Miscellaneous Publishers</td>
<td>2,000</td>
<td>250</td>
<td>.4</td>
</tr>
<tr>
<td>Commercial Printing</td>
<td>22,400</td>
<td>3,800</td>
<td>1.0</td>
</tr>
<tr>
<td>In-Plant Publishing</td>
<td>48,000</td>
<td>5,000</td>
<td>3.8</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>87,340</strong></td>
<td><strong>11,120</strong></td>
<td><strong>63.5</strong></td>
</tr>
</tbody>
</table>

Figure 7—The U.S. Market for Typesetting Systems

<table>
<thead>
<tr>
<th>Component</th>
<th>Systems Installed</th>
<th>Average Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typesetting Computer Systems</td>
<td>500</td>
<td>$100M</td>
</tr>
<tr>
<td>Optical Scanners</td>
<td>2,000</td>
<td>$60M</td>
</tr>
<tr>
<td>Editing Terminals</td>
<td>3,500</td>
<td>$335M</td>
</tr>
<tr>
<td>Phototypesetters</td>
<td>20,000</td>
<td>$500M</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$695M</strong></td>
<td><strong>$695M</strong></td>
</tr>
</tbody>
</table>

Figure 6

<table>
<thead>
<tr>
<th>Component</th>
<th>Unit Sales</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typesetting Computer Systems</td>
<td>1,500</td>
<td>$220M</td>
</tr>
<tr>
<td>Optical Scanners</td>
<td>2,000</td>
<td>$60M</td>
</tr>
<tr>
<td>Editing Terminals (including video display word processing equipment)</td>
<td>25,000</td>
<td>$250M</td>
</tr>
<tr>
<td>Phototypesetters</td>
<td>25,000</td>
<td>$400M</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$920M</strong></td>
<td><strong>$920M</strong></td>
</tr>
</tbody>
</table>

Figure 8—Computer Typesetting Systems Installed 1975

The most interesting non-system is the Newspaper Systems Development Group, a consortium of eight U.S. newspapers who have contracted with IBM’s Federal Systems Division to design and build a full page composition system capable of halftone composition, layout assistance, full page make-up and multi-column composition. It does everything we think a computer can do for a newspaper and a few things over which there is healthy debate. The project is over budget and at least two years behind schedule.

But the U.S. Printing and Publishing Industry has been willing to spend over $700 million on new technology in the past five years (Figure 6), primarily on phototypesetters and small production oriented computer systems. Newspapers have been the most aggressive sector here, and for good reason: payout on a typical newspaper system can be achieved in less than 18 months if the labor situation is right. Newspapers
While there are a great number of establishments—an astonishing number, really—less than 13 percent of these shops have more than 20 employees, a minimum number in my opinion to qualify the establishment as a potential site for a computer-based typesetting system. (Smaller shops will certainly buy phototypesetters and some may buy stand alone editing terminals, but few will be able to spend more than $30,000 on composition systems.) We have to ask a second question: How much typesetting is done? Fewer than a thousand newspapers, for example, set 55 million words a year while more than 3,800 commercial printing shops set only a million words. Analyzing the market this way suggests that of establishments with more than 20 employees, 970 daily newspapers set 86 percent of the type and therefore still constitute the giant share of the market. But the same 970 newspapers probably account for 650 of the typesetting systems already installed, leaving an untouched market of little more than 300.

Weekly newspapers, magazines and book publishers present perhaps a more interesting opportunity. There are 1,100 potential systems sites here and we estimate that only 150 systems have been installed. The opportunity for composition savings is not as great in these sectors and the publications are less sensitive to deadlines, therefore less demanding of speed. But major magazines have installed such equipment and are looking now for communications capability, and book publishers are trying to sort out the choice between word processing and communicating text editing.

The In-Plant market at the bottom of this figure is a tantalizing puzzle. Major corporations maintaining their own print shop will certainly buy typesetting systems. How medium-sized businesses ($25 million to $100 million) respond will be influenced dramatically by the shape of new word processing systems. Let me take the last few minutes to describe how I think this market will evolve. (Figure 8)

It is probable now that the sale of memory and communicating word processors will grow at a rate of 21 to 25 percent per year, reaching an installed population of 750,000 in 1980. The equipment will be used by secretaries and in administrative service centers to prepare reports, letters, legal documents, telephone directories and memorandums, many of which will be simultaneously stored in machine compatible media like magnetic card, cassette or floppy disk. This is the so-called word processing market.

We expect another development to occur at the same time: the general adoption of mini-computer based text editing and composition systems. By 1980 there will be as many as 1000 of these in medium to large businesses preparing formal reports, pamphlets, manuals, large directories and catalogs. This is what we refer to as the in-plant publishing market. It is obvious, I think, that in a short time these two independent components—the word processing world and the text editing world—will begin communicating. Material drafted on word processors will be composed in a central facility, long contracts stored in a central computer will be retrieved by the word processor to be revised. Product definitions will blur—“Is it an intelligent terminal or a communicating word processor.” Alert suppliers will identify and provide a full range of compatible products and we will hear advertising slogans that talk about a “total information system.”

And there is another development possible. In our own work evaluating and sometimes designing such systems we have seen cases where communications capability was required not only from the remote terminal to the central facility, but from the central facility to the main business data processing system and occasionally from facility to facility over packet switched communication networks or even dial-up lines.

What we are talking about here is a new business communications system that will provide a rapid, low cost alternative to the present process of typing, duplicating, mailing, distributing and filing. This has some important implications for those who would par-
ticipate in the market. It means message switching, privileged access, hierarchical organization of files and cross indexing, all this in addition to the communications and support systems.

For years we have used the phrase data processing to mean the manipulation of measurements. With the rapid introduction of computers to typesetting, text editing and word processing we are beginning to process ideas, to gather and select them, store and retrieve them, to format and present them in a way that will enhance their meaning. The associational structure of ideas and messages differs profoundly from the structure of numbers and we will need new strategies for storage and retrieval. The typography of an idea is no less important than the design of a complex statistical report, so we will need new commands, perhaps a new composition language. But the opportunity is enormous and the rewards for both business and society are rich. As computers become smaller, cheaper and more powerful we have the chance to use them for more than the measurement of work accomplished, we have a chance to move them out to the work site itself to speed and clarify the communications on which that work is based.
The computer as a tool in the processing of text for periodical publications

by WILLIAM J. HAMMOND
Publisher's Compu-Type Service, Inc.
New York, New York

ABSTRACT

This paper traces the introduction and initial applications of computer technology into the magazine publishing industry. It reviews developments leading up to our present state-of-the-art. It continues with an overview of today's systems. And, it concludes with a preview of future advancements designed to enhance the publishing of periodicals.

Since its introduction into the typesetting process in the early 1960's, the computer has proven itself to be not just a useful tool, but an almost indispensable one, in all areas of the publishing field. The techniques of setting type photographically had been successfully mastered in the late 1950's. The marriage of the computer to phototypesetting freed both the typographer and the publisher from the limitations and the constraints of three dimensional "hot metal" typesetting.

The first functions assigned to computers were those involving "end of line" decisions resulting in the justification of a line of type. Justification is the process of making a line of type fit evenly within the specified line measure. The computer freed the keyboard operator from having to determine when and how to hyphenate words, and from calculating how much space to allow between words. Thus freed, the operator could concentrate on improving the speed with which he could convert a manuscript into coded paper tapes to be processed by the computer. A new, justified paper tape version of the story could then be typeset on a phototypesetting device.

In the late 1960's, automation of typesetting processes began to make real progress as a result of the introduction of minicomputers. The "minis" may not have been as fast at internal computations as their big brothers, but they were fast enough to perform the many repetitive calculations involved in the justification process tirelessly and accurately, and still keep up with the relatively slow speeds of input and output peripherals. The comparatively low cost of minicomputers permitted the widespread introduction of computerized typesetting systems into an industry composed largely of small shops.

Initial acceptances of computerized typesetting in the publishing field were in the newspaper, book and catalog fields. Newspapers were quick to capitalize on the speed advantages offered by computers while books and catalogs found justification in their use based upon reprocessability for republication purposes. Periodical publishing lagged behind in its use of the computer because of its one-time usage of typeset matter and the relatively high degree of alteration of text prior to publication.

The introduction of "minis" not only permitted the widespread use of computers in the industry, but allowed systems to be developed with sizable on-line mass memory and more sophisticated programs to handle storage and editing requirements as well. Let's look at editing for a moment. Editing is the process of making words suitable for public presentation. It does this by performing a number of functions such as: selecting, compiling, writing, rewriting, altering, and formatting. The purpose of editing is to bring about conformity with a set of standards to suit a particular purpose. These standards, generally set by the publisher, might include: clarity of meaning; accuracy of spelling and hyphenation; and an effective or pleasant appearance of the final printed product. Editing is, in fact, a process of manipulation; manipulation of text, of words, of data. And, data manipulation is a process that readily lends itself to the electronic technology of computer processing. It is only natural that computers should prove to be useful and valuable tools for facilitating the text editing process.

Today's computerized typesetting and text processing systems are a far cry from those of 10 years ago. A modern system will include one or more powerful minicomputers. It will accommodate a wide variety of input and output devices. It will employ large random access disc systems for on-line storage and video displays for on-line editing and updating of files. And, it will have data communications capabilities to handle remote input and output to remote printing and publishing locations.

The modern computerized typesetting and text processing system affords periodical publishers and editors a number of facilities to improve their functions.
First, and foremost, is control. For the first time in many instances, control of how, when, and where typesetting is done is in the hands of the periodical publisher. Consistency and accuracy in typographic specifications and style integrity can be guaranteed. A publisher no longer has to depend upon a particular shop, or more likely a specific individual, to interpret an editor’s marginal notes concerning style. A few simple control codes can cause the computer to recall preprogrammed rules concerning: the type faces and sizes to be used; the relationship of various elements of the story; rules for the usage of intraword spacing; specific rules for word hyphenation and exceptions to those rules, etc.

The physical layout of text columns on a page is easily controlled. A predetermined size and shape for any column of text can be given to the computer and it will mold the text to fit the desired configuration. Both regular and irregular shapes of blocks of text can be used, resulting in more artistically pleasing relationships between text and illustrations on a page. Entire pages can be composed electronically with all typographic elements assembled in their proper position, thus saving both time and cost of assembling the elements by hand.

The correction processes are greatly facilitated by changing only the words or the individual characters requiring change. Formerly, changing even one letter on a line necessitated resetting the entire line; changing a word often involved resetting several lines. One of the great dilemmas of a publication editor has finally been resolved. Often, an editor had to decide between making the most effective change from the point of view of clarity of meaning, which could result in massive resetting of type, or making a less effective change and avoiding costly and time consuming resetting. Since in today’s systems, the text stream and configuration of text columns operate independently, the editor no longer need hesitate to make the most effective changes.

We are living in the midst of what has often been referred to as an “information explosion”. The need to know is steadily growing, and timeliness is being equated with newsworthiness. This is as true of the monthly trade magazine as it is of the national news weekly. The modern computer system is satisfying this need too. It has permitted tightening of schedules to the degree that news stories can break within hours of the time the presses must roll and still make the edition. Turn-around times for both original typesetting and corrections are such that many news magazines have justified systems on that basis alone.

Freedom from fear was promised 200 years ago, but it has only recently found its way into periodical publishing. With a modern computer system incorporating a high speed data communications network, publishers of large volume national news weeklies have been able to print identical, accurate versions of the news in multiple printing locations. They can now enjoy the advantages of multiple distribution points without the fear that different printers may have slightly different versions of the news. Control is in the hands of the publication. All magazine publishers, at one time or another, know the fear of having to move from one printer to another. Invariably, the most traumatic part of the move is the disruption and chaos resulting from moving the typesetting operation. “That little old typesetter with the green eye shade is the only person in the world who knows what the editors want!” With an in-house, or nearby service bureau handling the typography, the transition can be made without having to change editorial practices or procedures. Possibly the greatest fear of a publisher is the sudden, unexpected loss of services of the printer. Again, with the service bureau at hand, the publication is free to locate and switch to a new printing facility on a moment’s notice without fear of major disruptions in its editorial process.

Control, freedom, and flexibility make the computer an invaluable tool in the periodical publishing industry. These factors have taken typography out of the manufacturing process and put it back into the publishing process where it started.

What of the future? What additional benefits can the publisher look to derive from the computer? Many of the future benefits of computer usage are on the drawing boards and in the laboratories right now. The periodical publisher can look forward to systems that not only generate his typography, but also produce his illustrations; thus giving him the ultimate in control of the pre-press processes. In the area of marketing and distribution, computers will permit the manufacturing of personalized editions of his magazine for each individual subscriber. Advertisers will be able to direct their messages to highly specific audiences getting maximum effective exposure for their advertising dollar. Through modeling techniques, publishers will be able to specify the optimum combination of components in an issue, or they will be able to predetermine which of a number of suppliers is best able to produce a particular product. Of course, data base publishing will come into more and more sophisticated use. Libraries of past issues can be easily retained on magnetic tape with possibilities opening up for indexing, abstracting, and reprinting in new formats. Finally, with the perfecting of graphics generating typesetters, micro-publishing will begin to make inroads into conventional publishing methods. Someday, maybe soon, magazines will be entirely produced on microfilm or transmitted electronically directly into the subscriber’s home.

If this sounds like “blue sky,” just reflect for a moment on how far electronic technology has progressed in the past ten years, or even in the past five years. As technology continues to make giant steps in progress, that “blue sky” will rapidly come down to the solid reality of ground level.
The integration of microfilm and the computer

by DENNIS R. NEARY and TERRENCE H. COYLE
Eastman Kodak Company
Rochester, New York

and

DON M. AVEDON
National Micrographic Association
Silver Spring, Maryland

ABSTRACT

A discussion of the nature of information and how data flows through a basic data processing system. The key functions of INPUT, PROCESSING, and OUTPUT are reviewed, with emphasis on data capture and output reporting. After a typical information system is described, the various places where micrographics fits are analyzed. Microfilm is shown as a complement to the modern information system. The basic micrographics concepts are discussed. Rotary and planetary cameras are explained, as well as the various microforms that they can be used to create. Coding techniques and retrieval equipment are reviewed, and the value of microfilm as a storage medium is explored.

The nature of the COM concept is explored to establish where and in what situations it may, or may not, be valuable. Cost justification is explored as well as systems benefits. COM technology is explained, and the future of COM will be projected. An exploration of the concept of capturing source documents in random sequence on microfilm and using a computer generated index to assist in retrieval. The actual hard wire interface of a computer and a microfilm display device will be discussed. The future of computer assisted retrieval will be projected. The integration of data capture and microfilm will also be discussed. A projection on the status of data processing five years from now, and how microfilm will fit in the information systems of the 1980's.

INFORMATION FLOW ANALYSIS

In analyzing any basic data processing system, the major operations can be divided into input, processing, and output. Data is abstracted and captured from source documents, converted to machine language, and electronically transferred to the central processing unit, which processes that raw data in conformity with a standard application program. As a result of that processing, master files are updated and maintained, transactions are reported, and current status is updated and reported. The reporting has been done historically on paper and more recently with video images on CRT terminals. A third output image that has gained prominence in the last few years is the micro-image, created through computer output microfilmers.

The data processing system has undergone an information explosion. Bulging files, misfiled papers, long searches through millions of manila folders in thousands of offices all over the world, reams and reams of computer output, and expensive, complex on-line systems have strangled the typical modern business. Let's take a closer look at a typical data processing system and see where the micro-image can begin to increase the effective utilization of the entire system.

Information is first abstracted and captured from the source document. Keypunching, key to tape, key to disc, on-line terminals, and optical scanning are the major ways in which information on the source document is converted to machine language and ultimately transmitted to the mainframe. After the information is fed into the computer, what remains is the staggering problem of sorting, filing, and controlling the ocean of source documents. See Figure 1.

Source documents can be categorized as bulk file (unit data) or folder file (co-related data). Generally, bulk file consists of one type of document, and retrieval requires manual search of a paper file for one or more of these documents. Examples include checks, sales slips, time cards, stock certification, and the like. Folder file describes those groups of source documents which contain several types of inter-related data and require merging or updating. Examples include loan application files, insurance policy files, order entry files, and so forth. A key point in the definition of folder file is that folder file information requires inter-active updating. When a folder file becomes a dead record,
or purged information, it is then considered bulk file information. Regardless of whether a source document is maintained in a bulk or folder file, the micro-image can provide two major benefits: space savings and file management.

**Space**

The benefits of space savings are important to any growing business.

1. **Opportunity Cost**—The Micro-Image provides the opportunity to use expensive floor space for generating profit, not storing files.
2. **File Location**—The Micro-Image condenses files so that they can be located near the end user.
3. **Duplication**—Too many duplicate copies of dead records are a problem in many office environments. The Micro-Image solves the problem. The user can retain one microfilm copy and create paper copies only when needed. If duplicate copies of the entire file are required for several users, they can be generated at lower cost, occupying less space.
4. **Expansiveness**—Files can become so expensive that they not only occupy valuable space, but become unmanageable with both on- and off-premises storage. The Micro-Image can provide a 98 percent space savings, and virtually eliminate the need to selectively purge paper files.

**FILE MANAGEMENT**

The benefits of file manageability are benefits affecting the company's lifeblood, the flow of needed information.

1. **Operating Costs**—Paper files are labor intensive, making for a waste of talent in an organization. The Micro-Image with automated filing and retrieval helps reallocate talent, saving money for the prospect.
2. **Accountability**—Files document business operations, providing the basis for answering customer, management, or interdepartmental inquiries. The Micro-Image provides audit trial control over files, and eliminates the national average of nearly six percent of misfiles and lost documents in the source document area. The Micro-Image provides file integrity.
3. **Manageability**—The Micro-Image organizes files so they are of manageable size and format. And it pulls these files together so that they can be easily co-related on request. The Micro-Image manages the file through automated information handling.
4. **Security**—Paper can be easily tampered with, stolen, or destroyed. The Micro-Image is discrete, easily and safely stored, and can be duplicated with extra copies kept off-premises.
5. **Convenience**—The Micro-Image assures that the record is there, or it simply does not exist. Records can be accessed faster and more easily with any of the available formats.
6. **Human Factors**—Paper is essential. But after a message or order has been communicated and documented on paper, the manual filing process remains. Automated Micro-Image systems can also provide job enrichment to reduce employee turnover.

Microfilming of the source document can take place either prior to or subsequent to data capture. Microfilming can be performed in such a manner that the
information is accessible through various techniques which will be described later in this report. However, one approach which is gaining prominence in the recent past is the filming of the source documents in random order, and the assignment of a film address, a roll and frame number. This film address is captured with the other information passed on to the computer which in turn creates an index to the address by any number of parameters which the end user might require. This index can be reported out on paper, or can be maintained on-line, and allows the user to access information from a random film file with the assistance of the computer generated index.

After the information is processed and master files are updated, the computer produces a large number of reports. These reports are typically printed out on an impact printer on paper. Another approach is to maintain the report information on-line in a direct-access storage device and reference the information selectively through a computer terminal. The disadvantage of printing on paper is the tremendous time it takes—approximately 1,000 to 2,000 pages per hour—and the cost of paper and printing, and the problem of maintaining large volumes of paper reports for information retrieval.

The disadvantage of the on-line approach is the tremendous cost to store all the information in the direct-access storage device, transmit the information through a telecommunications system to the terminal, and the software to control the entire teleprocessing system.

A third alternative which has gained prominence in the past few years is a compromise between the two. Rather than maintain the information on-line at extreme cost, it is printed out; but not on expensive paper through a slow printer; but rather on inexpensive microfilm through a high-speed computer output microfilmmer. Through this technique, users are finding that they can provide much more information faster to a wider number of users at a much lower cost than they could with either paper output or on-line CRT display.

We have seen where microfilm is used to capture source documents prior to or subsequent to data capture. We also see the part that microfilm can play as an output medium instead of a paper or a CRT image. These two disciplines can also be integrated to provide a total information file which incorporates both paper, electronic image, and a micro-image. Consider source documents which are filmed and the address of that micro-image is passed through the computer to an output report. That output report is produced on a computer output microfilmmer, and serves as the index to retrieve the source documents on microfilm. Another group of source documents is captured in the same way, but their particular output report is printed out on hard copy. A third group of source documents are microfilmed, and their address is maintained on-line in a direct-access retrieval system. Now the user can retrieve source document information by either inquiring to a file with an on-line CRT which directs the user to the image on film, or the user can reference a hard copy report which provides direction to retrieval to the source document on film, or the user can reference an index on microfilm created by the COM, which also keys the end user back to the source document on film.

To take this inter-relationship one step further, a CRT terminal can be interfaced to a microfilm retrieval device so that when an inquiry is made to an on-line data base, the resultant micro-image of the source document can be electronically transmitted directly to the microfilm reader and the image subsequently displayed automatically. Systems of this type have been installed and operating for several years, and promise to proliferate in the near future. Now in order to better understand the advantages the micro-image can play in a data processing system, we will take a closer look at how microfilm is actually created and the various ways in which it can be retrieved. We will also look in more detail at the concept of computer output microfilming—how it is done and what advantages it provides over on-line access and paper printing. We will then review the advantages of computer assisted retrieval and project the future of the integration of microfilm and the computer over the next five years.

MICROGRAPHICS—INPUT AND RETRIEVAL

Subject

How microfilm is created and the various ways it can be retrieved.

Topics to be covered:
- Micrographic Concepts
- Rotary and Planetary Microfilmers
- Microforms
- Encoding Techniques
- Retrieval Techniques
- Microfilm as a Storage Medium

Theme

A basic microfilm system follows a parallel path to a basic data processing system. Our first speaker has presented a hypothetical data processing system consisting of input, processing and output. Since my topic follows a parallel path to this flow, I will use this same model to show "How Microfilm Is Created and Retrieved." See Figure 1.

While we are dealing with two distinct disciplines, there is some common ground. You are concerned with the creation, movement and use of "data records"; and we in the microfilm community are concerned with the creation, formatting and use of "film records."
For our purposes today, I will be discussing two types of "film records"—roll microfilm and microfiche. When I refer to roll film I am thinking of a reel of film 16 mm wide and either 100 feet or 215 feet long. And when we discuss microfiche I am thinking of a sheet of film 105 mm high by 148 mm wide. See Figure 4.

The parallel path I mentioned starts when we consider the data for the creation of both the "data record" and the "film record" is common—the same source document. You capture data magnetically and we capture the image photographically on film. The parallel continues until the end of the information path when you visually display "data records" and we visually display "microfilm records."

Let's begin our analogy by looking at input. You must extract pertinent data from source documents and enter it in your information system. You have a number of options—keypunch, key-to-tape, key-to-disk, on-line terminals and scanning. We are also interested in the input document, but from a different vantage point. We are not interested in extracting data, but in capturing the entire document image. To accomplish this task, there is one major option to consider—what type of microfilm camera do I use?

There are two basic categories of source document microfilmer—planetary and rotary. While there are numerous models of each, I will treat them in these two broad categories.

Planetary cameras are manual in operation. Documents to be microfilmed are placed on a copyboard and by depressing an exposure button the image is recorded onto microfilm. This procedure will be repeated from 3,000 to 4,000 times thus resulting in a completed roll of microfilm.

This type of camera is normally used when dealing with large documents or for precision microfilming jobs. Those of you from industrial or engineering firms may be familiar with engineering drawing applications utilizing 35 mm microfiches.

When dealing with normal business documents however, a rotary microfilm camera may be a better alternative. Rotary microfilmer can be used to greatly increase the speed at which microfilming takes place. This is possible because of automatic feeding devices which allow for creating microfilm images on the fly. In this instance, microfilming speeds of up to 615 documents per minute are easily achieved. Rotary microfilmers also offer other advantages, such as packing 15,000 check size images onto a 100-foot roll of microfilm and, at the same time, pictures can be taken of both the front and back of a document. These features, as well as automatic numbering and indexing, can be accomplished with no loss of throughput speed.

Now let's move from the input devices to the medium on which the data will be recorded. In our data processing model, data will be recorded on some magnetic medium—tape, disc, floppy disc, etc. While the medium is generally magnetic, the format does change. The same holds true for microfilm. While the medium is always constant, it can take different formats. See Figure 4.

16 mm roll film is still the most predominant format. Images are organized serially along the length of the film. We will explore how we locate an image in a few moments. However, if serial organization does not suit our needs, the roll of microfilm can be automatically cut up and added to a microfilm file folder called a jacket or film folio. This approach unitizes various pieces of data which have been received over a period of time. In this instance, the film file folder is identified by a descriptor such as account name or account number. The folder then becomes the repository for all documentation relating to that descriptor.

But let's go back to indexing. When you begin processing or moving "data records" you continually address (or index) the current location of that record. When microfilm is the medium, we must also be concerned with where images have been recorded. If a file is already serial we have no difficulty. A simple from-to label on the outside of the reel identifies the contents. The thousands of images on the roll can then be located by techniques such as codeline and odometer indexing. See Figure 4.

Codelines appear as solid bars between the microfilm images. Codelines equate to the parameter by which the file is organized. As the parameter changes to a higher value, the position of the codeline also advances. As film is advanced in a microfilm reader, the codelines appear as a solid bar on the reader screen. By establishing a scale on the side of the reader screen, film can be advanced until the codeline moves into the desired range of numbers.

Odometer indexing operates on much the same principle as a home audio tape deck. By noting the footage locations of where a given song starts, we can at any time go immediately to that piece. The same holds true with microfilm, except that we would be noting where a given range of numbers or names are located. Codeline and odometer indexing are generally used when dealing with general records retention applications. However, as we move into more dynamic environments the ground rules change considerably. In our data processing model, input is normally in a batch or random mode. If we are to microfilm these documents at random, we must then be able to establish film addresses for subsequent retrieval.

Our first speaker established one option for this type of retrieval. When a document is microfilmed it is identified by a "roll and frame number." In the microfilm community this approach is referred to as image control of item address. This film address is carried through your processing cycle and is available by interrogation of your master files, or on a report generated by your system. Once the microfilm address is identified, the proper roll of film is inserted in the microfilm.
retrieval device and the image control number is entered in the retrieval keyboard. The device then counts forward to display the information located at that address. See Figure 6.

A second alternative for random microfilming operates independent of computer processing for establishing the film location of a document. This technique involves the placement of a binary code on the microfilm adjacent to the image of the document. See Figure 4. The encoding takes place at the microfilm camera as the documents are being filmed. Retrieval is now accomplished by interrogating your film files by the descriptors which were encoded; example, account number, invoice number, etc. At this point we have followed our model system to the point of output. You are all well aware of your options for making information available to your users. Your options include automatically inserting images in a microfilm file and indexing techniques which I will cover briefly.

Earlier I discussed two microfilm formats—16 mm roll film and unitized microfilm file folders. Both of these formats are available when utilizing COM. The first roll film is available for serially organized files. The second, the unitized format, is accomplished by automatically inserting images in a microfilm file folder. But now I would like to introduce two formats available from COM—35 mm roll film and microfiche. See Figure 4.

35 mm roll film is similar to 16 mm roll film except that it is approximately two times wider. With 35 mm film, we can now print up to 16,000 images per 100 foot roll which is two and a half times our best alternative with 16 mm film. This is accomplished by printing five “tracks” of film images across the width of the film as compared to the two “tracks” available on 16 mm. The 35 mm five “track” format is particularly desirable where an extremely large data base or report is to be converted to microfilm via COM. Microfiche is a 105 mm by 148 mm sheet of film which can hold up to 288 data pages. Each microfiche is easily titled by report name and from-to report information. An index can also be created to locate the column and row location for each image on that particular fiche.

Microfiche is desirable for a number of reasons:

1. Large reports which will be used by many people are split into more manageable units. A person referencing a microfiche only removes up to 288 data pages from the file. By comparison, a person removing one roll of 35 mm film could be removing up to 16,000 data pages, thereby tying-up the file from any other user.
2. Small reports can be easily produced on one or two microfiche. Using 16 or 35 mm roll film would create many reels of film only partially filled which would have to be maintained in the microfilm file.
3. Perhaps the greatest advantage of microfiche is the low cost viewing devices which are available. Microfiche readers cost from $100 to $300, where a roll film reader could cost $1,000 to $2,000. I will come back to this point momentarily.

The introduction of COM into our model system also provides new opportunities for indexing our microfilm outputs.

Since the data to be recorded on microfilm is a product of your computer system, indexing can be accomplished under software control. Software has been designed by most COM vendors to:

1. Examine data fields to automatically generate code lines which correlate to the parameter by which the file is organized.
2. Examine data fields to automatically generate retrieval code used on Kodak Miracode equipment which corresponds to the descriptors known by the user of the file.
3. Abstract data fields to create an image control index to all data pages on a roll of film. This index page normally appears as the first image on a roll of film.
4. Abstract data fields to create an odometer index to key points in the given report.

An additional indexing technique not mentioned previously is also available when using COM. This technique,

5. examines data fields anywhere on a page and prints it anywhere on the microfilm image as a large eyeball code.

See Figure 4 for a complete review of all microfilming techniques.

The eyeball code is the technique used to generate the title and column heading information on microfiche. However, this technique can be used most effectively on 16 or 35 mm roll film applications to make these formats more retrievable for end users. In this case the large eyeball characters are normally printed at the bottom of a data page and correspond to the information by which the report is organized.

I fully realize that we have examined a great many concepts and new terminology in the past half hour. My intent has not been to confuse you, but to expose you to the tremendous flexibility of the microfilm medium. For you to use this information, it will be necessary for you to delve into the world of micrographics to determine which combination of microfilm cameras, microfilm formats, microfilm encoding techniques and retrieval options will best suit the needs of your organization. I will use an example to clarify this point.

A moment ago I said that one of the greatest advantages of using microfiche is the low cost of the
microfiche viewer. However, if we are dealing with an extremely large and centralized microfiche file this may not hold true. In this case it is conceivable that so many microfiche are created that a filing and retrieval problem is created. The result may be that the look-up time to retrieve an image is too long to satisfy a customer telephone inquiry. To satisfy this situation maybe a more automated roll microfilm format will better suit your needs. Indexing and retrieval options such as binary code and image control can be used to support roll film applications. While these options may cost many times more than microfiche viewers, the improved response time may well justify the differential. Only a thorough analysis of all available options will result in the most cost effective format and encoding technique. When we go back and examine our model data processing system in Figure 1, I hope that you would now agree that the "creation and retrieval of microfilm" do follow a parallel path. In many cases the design of effective source document microfilm systems will be dependent on your cooperation and implementation. But hopefully microfilm can do something for you as data processing people as well. This is the subject of our next speaker, so I will turn the program over to Mr. Don Avedon who will be discussing the COM concept and those situations where it may, or may not be valuable.

COMPUTER OUTPUT MICROFILM

We are a computer-dependent society. Thanks to computers, information can be stored, retrieved, manipulated and otherwise dealt with at electronic speeds. We know the information we need is there—and we need it fast!

For computer information to become human readable in permanent form, the output typically takes the form of paper. These massive amounts of computer-produced paper contain more information for use in our society than ever before . . . but in a form that is extremely bulky, cumbersome to retrieve, time-consuming to produce, expensive to store and costly to duplicate. As you know computers operate electronically at the speed of light in manipulating information, but in the printing out of the information, these electronic speeds are shackled to the slow speed of the paper-generating printer. While mechanical printer technology continues to produce higher and higher speed capabilities, accelerated paper printout today falls short of solving the demand for information when it's needed.

Consider the large, familiar accordion-folded computer printout sheets. To get at this information in this form, pages have to be leafed through, heavy volumes must be removed from shelves or file drawers . . . to disseminate information in this form means carrying, mailing, trucking or otherwise transporting bulky, cumbersome material at appreciable effort and expense. To store information in this form means considerable expenditure of space and money.

It does present a problem. We've all become so dependent on the computer that it is now part of our everyday lives both at work and at home. Paychecks, bills, reports, medical records, charge account files, business documentation, scientific data and virtually every type of information we need is now produced by the computer. The net result has been a data bottleneck, instead of a smooth, even flow of information in immediate response to today's high-speed demands. To give us our information, impact printers grind out heavy, bulky, expensive reams of paper that we humans must spend time, bursting, collating, filing, binding, shuffling through and spending lots of money on moving from place to place. Then we have to spend hours looking for what we want in a sea of paperwork. Computers have solved a lot of problems for us . . . but along the way they've created a lot of new ones!

Fortunately, a way has been found to solve the data bottleneck dilemma . . . a way that gives us information when and where it's needed. It is based on the meshing of two of man's most modern technologies—computers and micrographics. We have a three-letter word for it—COM—Computer Output Microfilm. Briefly, COM is the process of converting the output of a computer directly into a human-useable form that can be read by people and recorded on microfilm. Think of COM as THE OTHER WAY to make the conversion. The paper way consists of the computer's familiar accordion-folded page—which may run for hundreds of pages. The COM process reduces the same information to microfilm, achieving an enormous reduction in size. The result is a microform that offers many advantages which we will soon discuss. While speed output is not the major advantage COM provides, it is significant that COM outpaces impact printers and even the newer page printers.

Let's briefly review some speed comparisons and then move on to discuss the many advantages COM provides to its users. The line rate of the COM recorder is 32,000 lines per minute. The typical impact printer in widespread general use in the average Electronic Data Processing installation prints 2,000 lines per minute. As we mentioned, there are page printers, expected to gain increasing prominence which can function at 12,000 to 18,000 lines per minute and beyond. So you see, one of COM's advantages is a speed factor 16 times faster than the impact equipment used in today's typical systems and twice the speed of the new page printers.

So much for speed comparisons. COM's major advantages emerge when we compare COM-produced information with the same information in paper form. We will show how COM breaks the information bottleneck by making information human-useable—cheaper and faster.

Let's consider a typical computer printout of ap-
approximately 3,600 pages of 11 by 14-inch paper. The paper alone will cost about $35 and will weigh nearly 40 pounds, filling a large carton. If you want to send it coast-to-coast, you have to spend approximately $11. If you COM-process the same amount of data into microfiche, it will weigh only two ounces and will consist of fourteen 4 by 6-inch microforms which can be mailed FIRST CLASS (and will actually travel as Air Mail) for less than fifty cents. Besides this, the materials cost for the microfiche is under two dollars. COM-produced information in such microforms as cassettes, rolls or cartridges will, of course, reflect reductions in comparison with our carton of paper. See Figure 2.

We have talked about two advantages of COM: much greater speed and easier distribution. Before we go on to other advantages, let's pause and briefly examine the process by which data from a computer is converted into human-readable form and recorded on microfilm. Although the conversion process can initiate either directly on-line from the computer, it is usually done off-line from a magnetic tape unit for reasons which include not tying up mainframe time and equipment.

From the computer or the magnetic tape unit, the electronic signals that comprise the computer's data move to the COM recorder. In the most widely used method, the signals are reformed to permit their processing through a cathode-ray tube . . . a CRT . . . where they become human-readable light images which are microfilmed by a camera built into the film handling section. See Figure 3.

The exposed film from the COM recorder now goes through processing. It is usually standard procedure at this point to make a security copy of the information to guard against loss or damage. See Figure 4. Let's look at the aspects of COM processing. For one thing, the system may be set to produce direct positives—that is black-on-white—or reverse forms in which the text or other printed matter appears as white characters against a black surface. Doing this is simply a matter of photographic chemistry. Micrographics specialists maintain that many microforms achieve higher visibility when viewed "in reverse," but a positive black-on-white is better for producing hardcopy.
The next step is duplicating, where one of COM's major benefits becomes visible. With COM, once the master is produced, as many duplicate microforms may be made as needed, from one to hundreds. Instead of activating the computer or off-line tape unit for each set of duplicates required, the task is done by the COM duplicator at a fraction of the cost, and without tying up expensive EDP equipment time.

With COM, the user has a number of options in duplication. The most popular and widely used form typically produced at this stage is the familiar 4 by 6-inch microfiche which originates from 105 mm rolls. The COM user, however, may also select 16 mm rolls for loading into cartridges or cassettes, and another form—35 mm rolls—which are the most frequently used in tab size cards. And so, here you see another feature of COM systems—versatility. Many microforms for many uses! An insurance firm which needs to send data to 1,000 agents would use that many microfiches. A government agency might require a series of cartridges to keep track of personnel records . . . an engineering company can COM-produce its drawings on 35 mm film for mounting on aperture cards . . . and would generate that microform . . . it's all up to the user . . . and COM has the versatility to meet the need.

Now we are at the stage where the COM-produced microforms become the actual working tools for which they were designed. Distribution becomes simple, compared with handling and shipping bulky printed paper. The microforms can be mailed as easily as letters, saving time and expense. In filing, thousands of documents can be kept in a fraction of the space required by paper files . . . and in retrieval, information can be located instantly where it might take hours and sometimes days to hunt through unwieldy volumes of paper.

Now with COM-generated microforms in hand, we can gain access to this information with a microform reader. Selection of a reader is simply a matter of matching the reader to the microform system being used. There are compact, portable readers powered by house current or by self-contained batteries—even some that operate from the cigarette lighter outlet in your car. Most offices make use of desktop readers for personalized use, while many firms maintain free-standing readers for shared use within a group or department—all to insure speedy access to needed information.

And today, there are reader-printers, dual-function units that not only display all of your microform data, but also permit you to make hardcopy printouts when desired. Made at the touch of a button, these printouts may be produced in all standard sizes, including the computer-format size of 11 by 14 inches—all at the same high speed.

Enlarger printers are now in use which print out as many as 2,400 copies an hour from microforms. This speed factor adds high-volume production to their many other advantages and, of course, the over-all COM process has eliminated the need to first print out high volumes of paper or to make multiple printout
runs to obtain multiple copies. The ease in retrieving COM-generated information we spoke of earlier can be achieved by many systems now available. There are completely automatic systems which retrieve a microform at the touch of a few keys on a control console. For these automated systems using continuous roll microforms in cartridges or cassettes, COM methods are available to locate the desired images from coding placed directly on the film, an invaluable aid for rapid data retrieval. The user may choose among the various retrieval codes previously discussed.

Let's just pause here and review COM's major benefits.

Speed—Two million lines an hour as compared with 120,000 for impact printing.  
Economy—Material costs drop to 20% of paper printouts.  
Distribution—Pennies instead of dollars every time you must handle, transport, ship or mail—all with a speed of movement impossible with the former bulk and weight.  
Retrieval—Data location in seconds instead of minutes or hours.  
Storage—Reduced to less than 1% the space required by paper.  
Copy—Hard copies of any microfilmed information on demand in selection of paper sizes, in a matter of seconds.

These advantages have elevated COM into increasingly prominent use in just about every area of business and commerce. Typical COM business applications using alphanumeric equipment include microfilmed parts catalogs, customer/employee/vendor lists, financial records, transportation schedules, accounts receivable/payable, inventories, name/address/account number lists—and general business and governmental documents.

In many applications, the system user requires information to be generated in a special format. When alphanumeric information and forms must be combined COM saves both time and money. With computer impact printers this means that the paper must first be imprinted in order to receive the alphanumeric output. COM eliminates this need by simultaneously generating the column headings, lines, charts, graphs or other special artwork, and blends this with the alphanumeric information in the format that is wanted as the system is receiving the computer output.

Now, COM graphics recorders are available, systems that combine the alphanumeric capability with an unprecedented ability to handle graphics for such applications as scientific, engineering and business needs. This equipment is ideal for visualizing computer output in the form of bar charts, graphs, maps and diagrams. Progressing well beyond their function as high-speed substitutes for impact printers, COM systems are in use that are complete graphic arts recorders. They are able to produce high-quality graphics in any form that can be generated by the computer. COM-oriented graphic arts recorders are now producing engineering plots, line drawings, schematic and wiring diagrams, three-dimensional drawings, color charts, and even pictures, photo composition and movies.

COM applications are virtually limitless. It almost seems that each time a computer specialist and a micrographics specialist meet another COM use develops. Government agencies using COM include the Social Security Administration, which is probably the world's largest user of micrographics, the Internal Revenue Service and many other entities at the Federal, State and Municipal level.

Here, in summary, is the story of COM . . . the story of the solution to a paradox . . . computers capable of providing us with so much needed information can now function free of the data bottleneck that impact-printing imposes. COM systems are now at work serving all of us with the vital information we need in our business and personal activities . . . and the number of systems is growing.

**COMPUTER ASSISTED RETRIEVAL**

We have seen now the micro-image fits in the data processing system. Microfilm is used to capture source documents and it is used for output reports. The two technologies, data processing and micrographics, can be combined to provide in extremely powerful information storage and retrieval system.

Source documents are filmed in random order and assigned a sequential roll and frame number, a film address. The address is captured along with the data normally abstracted from the document, and processed by the computer. A cross index to the source document by one or more search parameters is created and either stored on-line, or printed out on paper or microfilm thru a COM. See Figure 5. When the index is stored on-line, it can be accessed via a remote terminal which is, in turn, connected to a microfilm display terminal. The user can now request dynamic information from a data base and, thru the index, retrieve source documents to support the data base information automatically.

**THE FUTURE**

Most people in the data processing community generally agree that we have progressed thru three generations of computers. Some feel that the fourth will be characterized by the creation and effective utilization of the data base concept. The fifth generation will provide hardware and software that will allow the ultimate user to comfortably interact with the data base.
Figure 5

The trend toward large centralized computing systems with extensive data bases appears to be supported by the recent introduction of IBM's 3850 Mass Storage System, an automated tape cartridge library that provides up to 472 billion bytes of memory on line. In order to maximize the effectiveness of these massive systems, efforts will be made in the next five years to distribute processing capability and provide information at decentralized offices for convenience, improved service and reduced operating expense. One of the keys to the success of providing massive computer power to geographically dispersed users lies in the technologically complex area of telecommunications.

Although it is fair to say that problems in telecommunications have been a stumbling block to date, there have been some recent events that seem to project a brighter tomorrow. IBM, Aetna and Comsat have announced a joint venture to provide satellite communications service. Technological advances in the field of Packet Network Switching promise to cut the cost of data transmission in half and several independent networks are already in service. The explosion of microprocessor technology will lead to more intelligent computer terminals and by 1980, many of the inconsistencies that presently impede the progress of telecommunications will be resolved, primarily with the emergence of IBM's SDLC standard. By 1980, we should have a new generation of mainframes which contain millions of bytes of MOS internal storage and billions of bytes of auxiliary disk storage. Although bubble memory and charged-coupled devices also promise to provide vast amounts of inexpensive auxiliary storage, their universal acceptance is still over five years away. On the other hand, the next five years will see significant improvements in MOS and disk storage. We will also continue to strive toward the concept of one massive integrated data base which will be maintained on-line to support a variety of batch tasks as well as interactive inquiry. However, it is doubtful that this utopian approach will be achieved by 1980. Real problems in file protection, contention and logic still remain to be solved. In practice, we will continue to employ, over the next five years, several data bases which are connected primarily by program logic.

There will be a proliferation of a variety of intelligent terminals to provide access to these data bases. Data Collection techniques will also improve as the flexible diskette, optical scanners and electronic funds transfer all mature. There are, however, still some problems. Even though auxiliary storage is decreasing in unit cost, we seem to have a multiple propensity to consume all available storage. After we load in the extensive data bases of the future along with a file of current records, we will still have to find a place to store documents, archival records, and reports of the processed transactions. The micro-image can play an extremely important part in satisfying that need.

The speed of output also continues to retard effective utilization of the computer system. Impact printing is not expected to progress to any great extent. However, non-impact printing should continue to enjoy technological advances that will increase its current capability. Computer output microfilming in particular will adapt to the computer environment and provide more convenient output at reduced cost. This trend in COM will further stimulate the integration of microfilm and the computer.

Most large computer installations and many moderate and small installations have already accepted COM as a viable solution to the expensive cost of paper forms and the bottleneck of impact printing. They are processing transactions, updating the data base, and reporting the transactions on microfilm thru COM. Retrieving the recorded transaction still presents a problem. A potential solution is to use the data base as an indicator of the address of the transactions on microfilm. Thus, the on-line interactive system will use the data base to assist in the retrieval of the micro-image. The computer will also serve to temporarily update the micro-image until it can be re-created.

And what about all those source documents? A discernible trend has already begun. That is to capture the source document on microfilm in random order and assign it a film address, a roll and frame number. This address becomes another element of collected data and is stored in the data base or is processed and incorporated in an output report that serves as an index to the original source document.

The problem today is that it is usually a two step
operation, microfilm and data capture. But the philo-
sophy of synergistic design will surely prevail and by
1980 we can expect hybrid, if not integrated devices
that simultaneously capture the data for the main-
frame and capture the document on a micro-image.
Both the data and the document will be forever con-
nected by the film address. Already today, high speed
scanners are available with a microfilm camera option.

The discussion thus far, has been oversimplified.
There are a multitude of variables and alternatives that
will produce many variations of the basic theme. Will
the computer be a mainframe, or a minicomputer, or a
timeshare service? Will the inquiry be batch processed
or interactive? Will the user be local or remote? Will
he use a CRT, printer, audio response, or something
else? Will the input camera be rotary or planetary,
and at what reduction ratio? Will the micro-image
medium be roll, microfiche, jackets, aperture cards or
something new? All of these alternatives will provide
tremendous flexibility in designing information sys-
tems to satisfy the needs of tomorrow.

And so what about the question of when to use micro-
film versus on-line interactive terminals versus some
other alternative? Well, that answer will be dictated
by the specifics of each individual application. As we
progress, trends will evolve. The integration of micro-
film and the computer already has established a viable
means to store, retrieve and update source documents
in such vertical market applications as law enforce-
ment, order entry, accounts payable and scientific files.
Computer assisted retrieval is also currently being
used on such computer output files as telephone infor-
mation, financial statements, inventory control reports
and computer generated accounting documents. The
number of source document and computer report ap-
plications will continue to grow and the scope of their
coverage will also expand.

The integration of microfilm and the computer will
take time, but it has already begun. The next few years
will see several vendors testing the market with hybrid
systems that interface computer terminals to micro-
film retrieval displays. The terminals will, in turn, be
connected to a mainframe, or a mini computer, or a
timeshare service. By 1980, integrated devices should
evolve that combine the functions of the terminal and
microfilm reader. In fact, some are already available
today. Data capture microfilmers should also be a
reality by 1980, including both the manual and auto-
mated high speed variety.

The most significant challenge will be in the areas
of system design, software, and file maintenance. Soft-
ware will be required to generate screen formats to
make the terminal easy to use for data collection and
retrieval.

Search routines will need to include and/or logic
with range capability, as well as the ability to handle
update messages. Because of the size of the average
file, unique techniques will be required to pack millions
of characters into a minimum of storage. As these
challenges are met, computer assisted micro-image
retrieval systems will flourish.

What the computer and microfilm industries need
most now is mutual understanding. They need to
realize that they are not competing with one another,
and that if they are successful in getting it together,
we will all benefit.

The integration of microfilm and the computer will
require the cross-training of the EDP and Micrographic
communities. This training, in the form of seminars,
lectures, formal curriculum and correspondence type
courses could be provided by the various EDP and
Micrographic vendors. In fact, some commendable
work in this area has already been done and has been
well received. Now what is needed is a cooperative
effort on the part of the various national organizations
representing these two disciplines. When workshops
on data processing techniques are delivered at micro-
film conventions and micrographic seminars are offered
at data processing meetings, we will be well on our
way to ultimate integration.
The AIDUS system—Automated capture, update and republication of maintenance manuals

by ARNOLD K. GRIFFITH
Information International, Inc.
Culver City, California

ABSTRACT

The AIDUS system is a hardware/software configuration which reads printed aircraft maintenance manuals, provides computer text-editing facilities for their update, and automatically republishes them onto microfilm. Page layout information is entered by a human operator at a digitizer tablet. This information is used to guide an OCR system in the capture of the text. The layout data is merged into the text output file, resulting in a computer readable file containing all the information from the original manuals except for the illustrations. The latter are captured photographically on a roll of 105mm film, which is keyed to the text/page-layout information file by means of bar codes on each frame. Updating of any aspect of the text or layout of the manual may be performed at CRT terminals using text editor programs. Illustrations may be modified by re-drawing, re-photographing, and splicing the resulting frames of film into the 105mm illustration film at the appropriate locations. After revision, the data file and 105mm illustration film are presented to the photocomposition portion of the system, which automatically produces a revised version of the manual, complete with optically merged illustrations and an index accommodating the revised pagination. This system, in use in an actual working environment, allows the update of technical manuals at about one sixth the cost, and with a fraction of the turnaround time, of previously used cut-and-paste methods.

Update and distribution

The motivation for the use of the AIDUS system is the increasing volume of maintenance manuals for out-of-production aircraft in use by the Navy. It is the Navy's responsibility to keep these manuals updated and to distribute updated versions to remote points on land and to ships at sea.

Today fewer new aircraft are developed and only a limited number of each are built. As a result, the Navy keeps them in service for a longer time. This trend to shorter production runs, longer in-service life and the increasing complexity of each new generation of aircraft has placed an ever-expanding burden on the Navy for accurate and timely documentation.

To illustrate the problem of increasing complexity faced by the Navy in publication activity, consider the A-3 aircraft built around 1955. It required 69,000 pages of documentation which was distributed across some 460 manuals. Ten years later the A-7 aircraft required 135,000 pages in over 1,000 manuals. Currently the Navy has approximately 17,000 technical manuals for the maintenance of these out-of-production aircraft, consisting of over one million pages and requiring continual update. This represents a formidable publication task and it is expected to more than double in scope over the next few years.

To facilitate distribution of these manuals, the Naval Air Rework Facility, the unit responsible for aircraft maintenance, implemented a prototype system by which the manuals would be reduced to microfilm and distributed in the form of extremely compact cartridges containing about 2700 pages each. Initially, 22 sets of paper manuals, each comprising 101 volumes, were removed from the A-4 assembly line and replaced by five microfilm reader/printer devices. The paper manuals having been updated and microfilmed, each set of manuals (101 volumes) was now contained on 16 microfilm cartridges, consisting of some 48,000 pages. After a one-year evaluation period, the system was funded and implemented for NAVAIR-wide use and designated MIARS (Maintenance Information Automated Retrieval System).

Having demonstrated the utility of the use of microfilm as an aid to distribution, the Navy turned to the problem of devising a means for the efficient and timely update of the contents of the manuals. After a three year study, it became apparent that this could be effectively accomplished by a total conversion of the text portions of the manuals, with update being accomplished by computer word-processing procedures.
THE ANATOMY OF AN AIRCRAFT MAINTENANCE MANUAL

A major problem in the conversion of the 17,000 maintenance manuals under Navy jurisdiction is the fact that virtually every page of every manual has a unique layout. This is due in part to the intrinsic complexity of the subject matter. In addition, the manuals were created by numerous different manufacturers over a span of more than a dozen years. Finally, the manuals have a variety of purposes ranging from the cataloging of parts to the description of maintenance procedures.

Simple text, constituting less than half of the total page area, is set in any of several dozen type faces and point sizes. This kind of text may be set on a page in two side-by-side columns, or be set in a single half width, one-third width, three-quarter width or full width column. In addition, the text contains figure and table references, whose location must be explicitly captured, since their location affects how the corresponding figures and tables may be laid out relative to the text.

Approximately one fourth of the total page area of these manuals consists of tables of one sort or another. The most prevalent type of table consists of a running parts list accompanied by a sequence of parts breakdown illustrations. These tables consist of up to a dozen tabulated columns, with a common heading on consecutive pages to identify the contents of each column. Although these tables continue for up to several dozen pages, their handling is complicated by the fact that the contents of the table must remain in step with the accompanying illustrations. In addition, the entries in certain columns of these tables are indented to varying degrees according to which level in a sub-assembly hierarchy the corresponding item belongs. Another type of table consists of a number of text passages or numerical data within a grid of ruled boxes. The structural constraints on such tables are minor. For example the heights of two boxes at different vertical positions within the table need not be the same, and a dividing rule need not extend all the way to either edge of the table. A third type of table, quite common in running text passages, consists of a number of lines of data tabulated in two or three columns.

More than one fourth of the manuals’ page area consists of illustrations. These may be halftones or line drawings, or both. They may occupy any position on a page, and either extend the full width of the page or occupy only one column. There may be several different illustrations on a single page with text interspersed.

Besides these major components, the manual pages also contain numerous items such as: running page titles at the top, or bottom, or both; page numbers; chapter titles; section titles; and centered captions over warning and caution notices.

INPUT PREPARATION

The data input process performed by the AIDUS system involves in the reduction of a paper manual to a computer readable data file containing all the text and page layout information, and a roll of 105mm film containing all the illustrations. Although the filming of the illustrations is a relatively simple one-step process, the capture of the text and layout information requires a series of steps. This series of steps may be divided into two phases, the first of which is termed the “Data Preparation” phase.

The first step in the “Data Preparation” phase is the punching of registration holes in the original pages. These holes are punched in a fixed relation to the material printed on the page, and are necessary for controlling the alignment of the text during reading, as well as the alignment of the illustrations during filming. In addition, unique serial numbers are printed on each page. Since each page is processed as a separate unit, three numbers are used as a common identifier for the various files created as the page passes through the successive stages of the data input process. The numbers are used to identify the illustrations as well. When a set of illustrations from a particular page is captured on a frame of 105mm film, it is this number which is bar coded at the edge of the frame.

The next step of the data input phase is the filming of all pages containing text on a 35mm planetary camera. GRAFIX I, the OCR component of the AIDUS system, reads only from microfilm, so that paper documents must be filmed prior to being read. There are, in general, several important advantages to this approach. The most important factor is that pages of any size may be optically reduced to a common film size, making page advance and page finding—the first, crucial stem in the recognition process—fast and accurate. Moreover, measuring the transmitted light passed through the film achieves a more consistent image quality than can be obtained by reflected light from the original paper documents. Another advantage is that the data can be filmed at the originating site and checked for completeness, avoiding the risk of lost or damaged material.

The final step of data preparation is called the “Page Descriptor Entry” procedure, and involves the entering of page layout data by means of a data tablet. Prior to the actual entry, a page is marked up by an editor to record which fonts are present, as well as to resolve certain unusual or ambiguous situations which might slow down the data tablet operator. After markup, the page is aligned on registration pins on the data tablet by means of the holes previously punched in the pages. The data tablet operator enters geometric information by touching various points on the page with the data tablet stylus. Logical information is entered by touching one or another of a “menu” of captioned boxes fixed permanently to the data tablet.
next to the page. For example, the page number is entered by touching, in order, the various boxes on the menu which contain the digits of the page number. The location of the text is entered by touching a box marked “text” on the menu, and touching the upper left and lower right corners of the actual blocks of text on the page. The layout of the various types of tables is entered by a similar, but more elaborate process, as well as the locations of the tab stops, titles, locations of cautionary notes in the text, various fonts, and other features of the page. The entire Page Descriptor Entry process, exclusive of markup, requires on the average of less than a minute per page.

TEXT CONVERSION

The second phase of data capture consists in reading the text of the manual pages to produce an output file containing text and page layout information, and subjecting this file to a proof and edit cycle to insure its accuracy and completeness.

The reading of the text is carried out on the GRAFIX I, a commercial optical character recognition system. GRAFIX I consists of three components. The first of these is a high precision flying spot scanner. The second component is a medium-scale general purpose time-shared computer. The third component is a slave processor called the Binary Image Processor, which performs a wide range of inner-loop recognition and image processing tasks at very high speeds. GRAFIX I has been successfully employed for numerous other multi-font character recognition tasks, and even reads intermixed alphanumeric handprint.

Recognition of the individual characters of the text is performed by a mask-matching algorithm. Recognition is dependent on the successful location of the successive characters in a line of text. This is performed automatically, sometimes by rather complex character separation algorithms, once the appropriate location and angle of the baselines of the entire line of text has been established. This baseline is, in turn, found automatically, given the location and orientation of the entire paragraph or column of text. This latter information is supplied to the computer during the Page Descriptor Entry procedure. The output file contains the passages of text together with imbedded typographical commands, which incorporate all of the page layout information entered during the Page Descriptor Entry procedure. In general, the structure and form of the page layout information, as entered during the Page Descriptor Entry procedure, is quite different from the corresponding layout commands as they appear in the text. But the process of generating these latter commands is rather straightforward. Certain additional commands are put into the output stream based on factors determined during the character recognition process itself. One such is the command indicating the beginning of a paragraph, which is generated by detecting the level of indent, and from certain textual clues.

Besides the text and layout commands, the output file from OCR contains the images of every character, which the system did not recognize, imbedded in the text at the point at which the character occurs.

The second step in the data capture phase is to subject this output file to a “Reject Conversion” process whereby a new file is created identical to the previous one except that the appropriate ASCII codes replace the images of the unrecognizable characters. This procedure is performed by reading in the successive lines of text until one is found with a “rejected” character, displaying this line of text together with the image of the character at a CRT display terminal, and having the operator key in the correct identity of the character at a keyboard. By this process more than 8000 rejected characters may be identified per hour.

The third step attempts to automatically identify recognition errors in the text output file. In certain instances, a positive identification of an error may be made, such as in the case of a numeral “1” appearing in the middle of a lower case word, which almost certainly is actually a lower case “L”. These errors are automatically corrected. Another procedure, using a dictionary, flags seemingly unusual words as possibly containing letter substitutions.

The entire file, complete with embedded layout commands, is now printed out on a high speed lineprinter for proofreading against the original. When the dictionary procedure is used, words flagged by the dictionary process are underlined in the listing, and proofreading is largely a matter of examining these underlined words and checking for completeness. The layout commands are also subject to proofreading at this point, since the layout command language was specifically designed with this in mind. Proofreading is followed by an editing of the file at a CRT/keyboard station to incorporate any necessary modifications.

REPUBLICATION ONTO MICROFILM

At this point a manual is transferred to magnetic tape, which is mounted, together with the appropriate 105mm illustration film, onto the COMP80. The manual, including all text, tables, and illustrations, is automatically republished onto microfilm in a recomposed form, complete with illustrations and an index revised to accommodate the new pagination.

Simultaneously, a data tape is created which is virtually identical to the input tape except that page and line breaks are in the same places as in the version being written onto microfilm. This tape is transferred to the GRAFIX I system where it is converted to a slightly different tape form for archival purposes. This archival tape is stored as the digital version of the manual; and it is retained and retrieved at a later date when a revision of the manual is necessary.
The process of creating a file for input to the COMP80 includes the checking of all composition and format statements for validity and plausibility of their respective numerical arguments. In the event that a statement does not pass these tests, an error message is printed, the file in question is edited, and the process is repeated. Since each page of text is converted as a separate data file, the various files constituting a manual are appended in the course of creating this output tape. In addition, since in general several manuals are composed on a single film, the various constituent manuals are concatenated onto a single tape. The selection of manuals to be placed on a single film is based on factors of convenience, and on the restriction that the total number of pages per film must not exceed 2700.

Since output is composed and typeset automatically, hand stripping and pasteup are eliminated. The labor and time normally associated with book make-up operations is drastically reduced. Rather than a six to 18 month production cycle, manuals can be updated and distributed in as little as 60 days.

Another aspect of GRAFIX I/AIDUS output capability is that publication formats can now be standardized uniformly or modified for new distribution requirements. The updated Navy technical manuals will now have a uniform format as documents from a wide variety of sources are recomposed and republished. In addition, a proportionally spaced type font was designed for the microfilmed manuals which attempts to be compact but maximally readable with the MIARS film reader.

UPDATE FOR PUBLICATION OF REVISED VERSIONS

When it is necessary to revise a manual, its archival tape is read back into GRAFIX I disk storage and divided into its individual pages. These pages are edited, and the publication cycle described in the preceding section is repeated. In addition, any revised illustrations are re-filmed and spliced into the 105mm illustration film.

Editing is performed on a per-page basis by GRAFIX I's special editing software. Using on-line CRT/keyboard terminals, editors may work from either the terminal display, hardcopy printout, or composed film proofs of the converted text. Material to be updated can be accessed by manual, section, chapter, and page number. Updated portions on film may be identified by vertical change bars in the margins; previous change bars are automatically deleted in update runs.

Updating functions include text deletion, replacement or insertion; arbitrary layout changes, accomplished by modifying the page layout commands imbedded in the text; modification of tables including content and layout; illustration additions, deletions, replacements or size changes.

New or updated illustrations are filmed and merged with the illustration file previously filmed during Input Preparation. When a particular illustration is supplanted, its successor is assigned a number higher than any so far on the film, and this number is bar coded on the edge. The frame is spliced into the film in place of its predecessor. A simple change of reference number in the text file completes the process. New illustrations may be added by a similar procedure.

CONCLUSIONS

We have described the AIDUS system, which consists of an OCR device, CRT/keyboard facilities for performing text editing, and a photocomposition system. We have shown how, with a judicious combination of filming, human interaction, and optical character recognition, this system is capable of reducing the entire contents of complex technical manuals, including layout data, to an illustration merge film and a computer-processable data file. In this form, the manual may be easily revised and subsequently republished onto microfilm. We have described an actual working environment in which this system is used, namely the update of maintenance manuals for out-of-production Naval aircraft. In this environment, the system performs update and republication at a projected long-range cost of 15 percent of that of cut and paste methods previously employed and at a fraction of the turn-around time. This system appears to have a wide range of application such as parts catalogs, and maintenance and technical manuals of all sorts.

ACKNOWLEDGMENTS

Designers of the system include, besides the present author: Richard Martin, Russell Ham, Ron Traver, David Meredith, Carol Gillespie, and Dan Forsyth. Other contributions were made by: Steve Gray, Mary Strobel, Glen Williams, Nada Berger, and Norm Gilbreath.

REFERENCES

The CMU RT-CAD system—An innovative approach to computer aided design*

by DANIEL P. SIEWIOREK and MARIO R. BARBACCI
Carnegie-Mellon University
Pittsburgh, Pennsylvania

INTRODUCTION AND MOTIVATION

As technology has evolved the primitive components available to a digital system designer have changed dramatically. Twenty-five years ago the designer constructed systems out of circuit level components such as resistors and diodes. Subsequently switching circuit level components, as represented by gates and flip-flops, became available as small scale integration (SSI) components. With the introduction of medium scale integration (MSI) register transfer level components appeared: arithmetic and logic units, registers, shift registers, etc. The advent of large scale integration (LSI) has made memories and even processors primitive components from which systems are designed. Two trends can be observed from this technological evolution: (1) primitive components continue to increase in complexity and (2) the rate of introduction of new components continues to increase.

In response to the first trend, designers have been limiting their excursions into switching circuit level design to only small portions of the system (e.g., bus controllers, etc.). In some register transfer level module sets, these excursions have been completely eliminated.

Because of the second trend, rapid technology evolution, there is a need to shorten the delay time between the introduction of a technology and its effective use in new computing systems. The design process must, therefore, be accelerated if the potentiality of the improving technology is to be realized.

This paper describes a set of design programs, the RT-CAD system, developed at Carnegie-Mellon. The ultimate goal is to minimize the effect of changing technology by building a Computer Aided Design System that implements a technology-relative design process.

* This work was supported by the National Science Foundation under Grant GJ-32783X.

OVERVIEW OF THE AUTOMATIC DESIGN PROCESS

Given the complexity of a digital system, designers have sought to develop automatic means to reduce the cost and time of the design process. The objective was to relieve engineers of repetitive, time consuming tasks such as:

1. The generation of detailed design information (gate and chip types, etc.)
2. The control of changes in the design documents
3. The checking of the system for electrical, logical, and physical compatibility (fan-out limits, etc.)
4. The generation of detailed manufacturing information (chip placement, board layout, etc.)

This early view of design automation limited itself to filling the gap between the low-level design specifications and the manufacturing data. Behavioral specifications were in the form of Boolean equations and the design programs translated them into their equivalent logic diagrams and wiring lists. Most of the synthesis algorithms at this level dealt with the problem of reducing or simplifying the Boolean equations.

Subsequent efforts were directed towards a system capable of accepting a higher level of behavioral description, although still oriented towards a gate level implementation.

Current design automation effort is shifting from implementation in terms of the switching circuit level to implementation in terms of the Register Transfer Level (Although the components have changed, the specification of fabrication information such as PC board layout, chip placement, wiring and cabling, still has to be performed in more or less the same fashion as before). Register Transfer level simulators have preceded this trend by several years. The closeness of
RT level descriptions to conventional programming accounts for this early success. Register Transfer level descriptions are easy to transliterate into executable programs in a conventional programming language (e.g., FORTRAN, Algol, etc.), thus providing inexpensive and fast simulation (although in many cases RT languages are compiled directly). Register Transfer level synthesis algorithms have been less successful. A few programs have been developed that take an RT level description as input and compile it directly into a known set of RT level hardware modules (for instance, CHARTWARE® and AHPL®). Figure 1 depicts a typical RT design automation system. The RT level description serves as input to several software modules. Syntax checking insures a well formed description. Static checking attempts to locate logical design errors (such as deadlocks, redundancy, etc.). The simulator is used to debug the design dynamically. Finally, the description is cast into hardware via the physical design programs.

The essential feature lacking in conventional RT Design Automation (DA) systems, and DA systems in general, is the exploitation of alternative implementations derived from the initial behavioral specification. Consider the augmented DA system depicted in Figure 2. The inputs are the RT level description and designer given constraints. The output is the specification/simulation of the hardware that attempts to optimize the system according to the design constraints. By allowing the description of alternative module sets the system can perform design relative to technology thus speeding up the incorporation of new technology into the design process. Also, such a system will allow experimentation with multiple module sets, each tailored to a specific class of problems. The system would also facilitate the design of the module sets themselves. Since the system operates on a symbolic description of the modules, a non-existing module set can be fed to the system for experimentation purposes. Such experiments will point out the advantages and disadvantages of a proposed module set.

At this point it would be instructive to describe the order in which the DA programs are typically used in the design process. This will serve to place subsequent discussions in perspective. Given a computational task, there are usually several algorithms that can be employed. The algorithm that is selected by the designer is described to the design automation system (Figure 3) and placed in a data base. Subsequently all design automation programs will use this data base. A high level simulator can execute from the data base to facilitate user debugging of the initial description.

Next some evaluation and reshaping of the algorithm is undertaken. Analysis tools have been developed to check the algorithm for well-formness (e.g., deadlock conditions, etc.). Perturbations of the basic algorithm can also be attempted such as: series-to-parallel transformations, replacing loop counters by wired-in control, and using table look-up in lieu of computing the value of functions. Thus attempts are made to first
bind those design decisions with global implications. While these perturbations can be performed independent of the physical design, the evaluation of their ultimate desirability may depend upon the module set used to implement the final, physical design.

Finally, the actual physical design is performed in terms of RT level modules. The module set can be selected from a library of module sets or a user described set. At this level several forms of allocation variations are encountered:

- Registers. Determine the allocation of the abstract variables to registers and memory.
- Data operators. Determine the number of operators of each type in the design.
- Control. Select control schema from among unary state encoding, binary state encoding, microprogram control, etc.
- Bus-Link clustering. Many RT designs start with a set of registers for variables and interconnect them with links to operators (add, shift, multiply, etc.). After a point the interconnections between certain registers and operators become numerous enough to warrant replacement by a bus.
- Operator interconnection. The interconnection of operators has been shown to have a significant effect on the test generation effort required for the physical implementation.\(^\text{11}\)

The signal level design verifier can be used to analyze the intermodular signal relationships in proposed module sets. Even well-established module sets have exhibited deadlock behavior in what appear to be straightforward interconnections.\(^\text{16}\)

A first version of the above system has been implemented at Carnegie-Mellon and is shown in Figure 4. The behavioral specifications of the system to be designed are provided in terms of the ISP language.\(^\text{12}\) The compiler produces an “object”\(^\text{*}\) program which is then loaded into the data base and manipulated by the different design programs.

The next six sections will treat the applications programs in detail. The third section describes EXPL, a module independent design program that examines series-parallel variations in the original algorithm. The fourth section presents the physical allocators for two existing RT module sets—RTMs and Macro-modules (MM). The fifth section discusses the heuristics used by EXPL to explore the design space. Sample design spaces, examples of the application of the heuristics, and some observations are presented in the sixth section. Design verification is discussed in the seventh section. The last section concludes the presentation of the existing system by briefly outlining the remaining application programs.

**AUTOMATIC DESIGN SPACE EXPLORATION**

EXPL\(^*\) takes as input the object code produced by the ISP compiler, together with a set of user given

---

\(^{*}\) The compiler produces an “object” program in terms of a set of Register Transfer level primitive operations. This program appears in the form of an executable BLISS program where each Register Transfer operation is represented by a call to a user-provided subroutine. By changing the set of subroutines, the compiler can support many diverse activities. The creation of the data base is, in fact, done by a specific set of subroutines. The compiler and the language are therefore independent of the applications. The uniform compiler output and the flexibility of the subroutine-call mechanism has simplified the interfacing to other application programs.

---

From the collection of the Computer History Museum (www.computerhistory.org)
speed/cost constraints/tradeoffs. The compiler output is used to generate a graph representation of the behavior of the system. Subsequently, various series-parallel and parallel-to-series transforms on the graph are attempted to establish a new design. Several alternative designs are generated and passed to module set evaluators which complete and evaluate the design in terms of its hardware module set. Using this evaluation and a set of heuristics, EXPL decides which solutions should be kept to generate other solutions by yet another application of the graph transformations.

Figure 5 is the ISP description of an 8-bit multiplier that will be used as a running example to illustrate various aspects of EXPL. The algorithm is a variant of the shift-and-add algorithm. The multiplier is in the P register and the multiplicand is in the MPD register and is assumed to occupy the leftmost 8 bits of the register. The product will be in the P register. The partial products are formed in the left hand side of the P register and shifted to their appropriated position in the final product. A counter, C, is used to keep track of the number of times the basic multiplication step has been performed. Additional details about the algorithm can be found in Reference 1.

The description begins with the specification of the label for the program (MULTIPLIER). Labels are used in ISP to identify activities so that they can be branched to, or used as subroutines.

The program itself is enclosed in parentheses, and consists of two parts. The declarations and the specification of the behavior. The former are specified as a list of individual component declarations (multiplicand, multiplier/product, and step counter), using the reserved identifiers DECLARE and ERALCED as brackets. The specification of the activities of the system is given as a list of two sequential steps. The first step (C<-8) initializes the counter and the second is given by a labelled (L1) block of activities. These consist of a sequence of three steps. The first one performs the basic multiplication operation; the second step decrements the counter; the third step tests the counter to see if the operation has been completed. If the value of the counter has not reached 0 then a jump to the label is indicated by using the label as an activity. If the counter is 0 then control flows out of the labelled statement and reaches the end of the program.

The basic multiplication operation is described using the DECODE control operation. It implements an n-way branch depending on the value of the expression following the operator. The alternative paths selected by this operation are given as a list using the “;” as delimiter. The first path (P<-P↑SRO 1) is selected if the value of the controlling expression (P<0>) is 0; the second path (P<- (P+MPD) ↑SRO 1) is selected if the value is 1. The operator ↑SRO represents a shift right inserting zeroes. The number of shifted positions is given by the second operand (in this case the integer 1).

Figure 6 shows the graph representation of the ISP description. The mapping from the ISP description to the graph form is apparent from the example. The system graph contains a unique entry point (the START operation) and a unique exit point (the STOP operation). In addition to these two operations, there can be five other types of operations in the graph model:

---branch, activates one of the output paths depending upon the value of some operand.

```
MULTIPLIER:=
   (declare
      MPD<15:0>;
      P<15:0>;
      C<15:0>
   eralced
   C<-8 next
L1:= ( (decode P<0> =>>
      P<-P↑SRO 1;
      P<- (P+MPD) ↑SRO 1) next
     C<-C-1 next
     (if C neq 0 =>>L1)
   )
```

Figure 5—The ISP description of the multiplier

Figure 6—Multiplier graph 0 (original)
—serial-merge, activates its output path when any of its input signals arrive.
—diverge, activates concurrently all of its output paths.
—parallel-merge, activates its output path when all of its input signals arrive.
—data-operation (other).

Examination of the graph for the multiplier example indicates several possible alternative designs. For instance, the computation of the loop count \((C-C-1)\) does not depend on the shifting and adding steps \((P\leftarrow P + SRO\ 1\) and \(P\leftarrow(P+MPD) + SRO\ 1)\); the two sets of operations do not have variables in common. Thus the decrement of the loop counter can be performed in parallel with the basic multiplication step, as shown in Figure 7.

The graph thus obtained shows that the testing of the loop counter, although independent of the multiplication steps, cannot be performed in parallel with the decrement of the loop count. This fact rules out a transformation similar to the one used previously. However, it is possible to insert the testing step in the same path as the decrement. This preserves the required ordering of the counter operations but now the testing is done concurrently with the multiplication step as shown in Figure 8.

The last graph represents an "optimal" speed implementation. We are not considering, at this point, specific module-set-dependent optimizations. For instance, register allocation in RTMs or data operator allocation in MMs. This type of optimization is left up to the individual technology-dependent evaluators.

It should also be noted that, in the example above, although it took two steps to arrive at the final design we could have taken the two counter operations (decrement and testing) as one group, and place the group in parallel with the basic multiplication step. In other words, we can achieve the same final result in one step by varying the size of the graph partitions.

These graph transformations have taken us from an original solution, to two additional design alternatives. Both represent an improvement in speed with respect to the initial design. The design space, explored automatically by EXPL, can be represented as a two di-

---

From the collection of the Computer History Museum (www.computerhistory.org)
Each design is represented by its cost and time coordinates, computed by the technology-dependent design evaluators.

The multiplier example was implemented using both RTM's and macromodules and the design spaces are shown in Figures 9 and 10. In both plots, point 0 represents the initial solution and points 1 and 2 represent the alternative designs. Arrows indicate the steps taken in the derivation of the alternatives.

The position of the alternative design in the design space is dictated by the evaluation of its cost and speed. These two parameters are measured in terms of a specific technology, using specialized evaluator routines, as described in the following section.

**MODULE SET EVALUATORS**

Given a candidate design, whether it is the initial graph or one obtained by a transformation, its cost and speed must be measured to ascertain its relative position in the design space. This position indicates the quality of a solution. The evaluation process is clearly dependent on the module set used and is currently performed by ad-hoc routines, independent of the graph transformation algorithms. This section describes the evaluation procedure used for RTM and Macromodule systems.

The candidate design is represented by a description of its behavior, i.e., a graph. In this representation the control and data operations are not bound to specific physical components. The evaluation is performed by applying a series of binding algorithms that map this (abstract) behavioral description into a physical description. This is the representation which is then evaluated. It should be pointed out that the design space we are dealing with is really an evaluation space, not a structural space. In other words, for each point in the evaluation space there may be more than one structure.

Different design spaces are explored by using either the RTM or Macromodule evaluator. The reason for this is inherent in the interconnections of the data and memory components allowed by each module set. Specifically, in the RTM set (Figure 11b) the inputs to the data operators are permanently connected to memory outputs (DMgpa). This means that some form of register allocation must be done to insure that at all times the proper operands are present at the input of the data operators. The RTM set also provides a common bus interconnection between the modules. This allows memories not directly connected to data operators to share the centralized DMgpa much as a computer's main memory shares the data operators connected to the central registers. In contrast, the Macromodule set represents a different style of design. Memory and data elements come in separate modules that are directly connected with data cables or links. Instead of sharing the data operators, each operand is bound to a register with only the necessary data operators connected to it (Figure 11a). The resulting Macromodule system has relatively expensive data operators and links for each operation, whereas an RTM system will share data operators and links (buses) with all of the operands.

Figure 12 illustrates how the concept of a design style is incorporated in the physical allocators of the RT-CAD system. RTM and Macromodules represent two differing register transfer level design styles:
central accumulator, and distributed data and memory, respectively. In addition, other design styles such as pipelining can be defined. An abstract system description is presented in graph form to one of the design style allocators. Each design style allocator has its own procedure for allocating operands to memories and data operations to operators. Temporary registers and extra register transfers might be added to the abstract design so that it will conform to physical design constraints. The result is a perturbed graph that reflects a specific design style and module set. The graph, which now represents a physical design, is evaluated using the costs and speeds given in the module set description. The results of the evaluation are then used by EXPL to produce an alternative design.

The allocators must be able to work interactively with the module independent subcomponents, designing and evaluating at varying levels of detail in the design process. At the style level, the allocator begins binding variables and operations to physical components using a set of allocation algorithms specific to the design style but general enough for any module set reflecting that design style. At the module set level, information specific to an individual module set is read from the module set description and used to help complete the allocation. In this way the allocator can function with several different module sets reflecting the same design style. An example of this can be found using the RTM set. Consider a module set similar to RTM except that the A and B registers of the DMgpa are symmetric. The allocation algorithms are very similar except for a few details such as not being able to produce $B-A$ in one register transfer in the real RTM (although $A-B$ is available).

The module set level of the allocation uses a set of templates to map the abstract operations in the graph into a module set specific structure. These templates, produced by the template generator from the module set description, act like macros in an assembly language. For each abstract data operation in the graph there is a set of templates that indicate, for example, the different groups of RTM operations that can be executed in order to achieve the desired effect. An operation like $A \leftarrow B + C$, where A, B, and C have been allocated to memories without data operators (central accumulator design style allocation) is mapped into a sequence of RTM primitive operations like: $DMgpa/A \leftarrow B$; $DMgpa/B \leftarrow C$; $A \leftarrow DMgpa/A + DMgpa/B$ (module set level allocation).

Templates of this nature are used by the Macromodule evaluator, although the nature of MMs allows more flexibility than RTMs. For instance, using the above RTM example, there are several ways of implementing the statement in MMs and the choice may be critical. With all variables allocated to registers and the necessary data operators connected between the registers (distributed data and memory design style allocation), the statement $A \leftarrow B + C$ can be implemented alternatively as $A \leftarrow A + B$; or $A \leftarrow A + C$. At $A \leftarrow A + B$, since at the module set level of allocation it is found that all data operations have the destination variables as one of the source variables. (A module set without this restriction could use the same design style allocator.)

This decision is critical and depends upon the data operators already connected to a given register (in this case, register A) or which will be connected in order to implement subsequent operations. If the operator $A \leftarrow A + C$ has already been placed in the stack of register A, the first option is clearly the one to adopt. On the other hand, if none of the operators exist then the template adopted depends upon the future uses of the operator, a decision based on a global analysis of the graph.

In addition, a module set may contain a data operator which cannot be described in a simple ISP statement. For instance, the RTM DMgpa can operate on two operands and then shift the result right in one register transfer. Operations such as this require that the allocator scan the graph for occurrences where the special operator can be used. Detecting these occurrences usually accounts for major optimizations in the final design. In fact, most good designers look for special features of module sets and try to utilize them wherever possible. The Macromodule allocator was written to search for special operations such as:

- Invariant bit mapping within an operand (field extraction, bit swapping, etc.)
- Comparing with constants,
- Comparing constants with bit fields,

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{module_set_relative_design_process.png}
\caption{Module set relative design process}
\end{figure}
Macromodules have special physical constructs that allow the above operations to be done easily. This specific information about the Macromodule set inserted at the module set level of the allocator allows the automatically generated designs to approach designs produced by a good designer.

Table I compares some sample designs produced by hand and by the RT-CAD system.

### HEURISTICS AND DESIGN SPACE

**TRADE-OFFS**

Due to the interaction between series/parallel transformations in EXPL it is a difficult task to formalize the optimization (improvement of alternative structures) as a mathematical optimization problem. The main difficulty is the fact that transformations apply to subgraphs of arbitrary size and, as a consequence, transformations in a given alternative structure may or may not be feasible or desirable in structures derived from it. It is also the case that new cases of transformations become feasible or desirable only after a specific sequence of transformations has been applied.

Two parameters will be used to describe the design space: The cost of the hardware involved and the operational time. The former is obtained by adding the costs of the components used in both the data and control structures. The latter is obtained from the average speed of the operations involved.

For a straight sequence of operations the time required is the sum of the individual times. In the presence of concurrent activities, the operation time is that of the longest (timewise) sequence.

When computing the times required by the alternative paths of a branch operation EXPL assumes, by default, that all such paths have equal probabilities of being executed (the probability being 1/n for n-way branches). This default can be overruled by the user by specifying the branching probabilities for the individual paths. The computation of the times required by the paths is then weighted by the branching probability associated with the path. The execution time is then the sum of these weighted path times.

The presence of cycles (loops) adds some complexity to the estimation of the operation time. In this case the level of nesting is assumed to be proportional to the number of times a loop is executed (i.e. the number of copies) is usually unknown. This specific information about the Macromodule set inserted at the module set level of the allocator allows the automatically generated designs to approach designs produced by a good designer.

The number of times a loop is executed (i.e. the number of copies) is usually unknown. This specific information about the Macromodule set inserted at the module set level of the allocator allows the automatically generated designs to approach designs produced by a good designer.

**TABLE I**

<table>
<thead>
<tr>
<th>Design</th>
<th>Hand Design</th>
<th>RT-CAD Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minicomputer</td>
<td>Cost: $6500</td>
<td>$6500</td>
</tr>
<tr>
<td></td>
<td>Time: 3.90usec</td>
<td>3.92usec</td>
</tr>
<tr>
<td>MSG</td>
<td>Cost: $6000</td>
<td>$7450</td>
</tr>
<tr>
<td></td>
<td>Time: 7.1usec</td>
<td>13.1usec</td>
</tr>
<tr>
<td>Multiplier</td>
<td>Cost: $3900</td>
<td>$3900</td>
</tr>
<tr>
<td></td>
<td>Time: 5.7usec</td>
<td>5.7usec</td>
</tr>
</tbody>
</table>

Assume that the primary objective is a reduction in time and that the designer wants a cost/speed trade-off of at least m dollars for each time unit saved. The objective could be something like: “the fastest alternative structure not costing more than x dollars”.

For simplicity, the subspace of acceptable solutions will be defined by a set of straight-line segments whose slopes reflect the objective functions. In the example above a single straight line, parallel to the cost axis, would be used to divide the space in two halves. Only those solutions that lie in the semispace containing the origin are considered acceptable. These solutions represent improvements along the design goal.

More complex constraints can be described by using lines of the form \( C = mT + b \), where m is a parameter indicating how many dollars the designer is willing to pay for each time unit saved (if time is the primary goal) or how many time units the designer is willing to sacrifice for each dollar saved (if cost is the objective). \( m \) is termed the cost/speed trade-off factor. An example, Figure 13, will clarify this description.

Assume that the primary objective is a reduction in time and that the designer wants a cost/speed trade-off of at most m dollars for each time unit improvement. Furthermore, assume that the original design is characterized by \( C_1 \) and \( T_1 \). The “acceptable trade-off” subspace would thus be delineated by two line segments: one parallel to the cost axis starting from
Figure 13—Design space reduction

(T1,C1) to (T1,0), and the other through (T1,C1) with slope -m. Further assume that by studying the control flow and data dependencies in this original structure, four transformations are available which yield four alternative solutions derived from the original one: A,B,C,D.

By dividing the space according to the trade-off lines alternatives B, C, and D can be rejected because their characteristics are not within the acceptable subspace (i.e., they take more time or the decrease in time costs too much). The alternative left, A, represents improvement in time while the cost to achieve the improvement is under the cost/speed trade-off threshold.

The process can now be applied to A in an identical manner. Design A is taken as the new initial solution and a new "acceptable trade-off" subspace is defined by a line segment (T2,C2) to (T2,0) and a line with slope -m through (T2,C2). Since in some cases more than one alternative can be left for further exploration, this process takes the form of a tree walk where the nodes represent alternative solutions and the edges are the transformations applied. In some instances, identical structures can be obtained by different sequences of transformations and the exploration of the design space is a graph-walking process. In any event, a path ends when no alternative solutions worth exploring can be reached from a given point. When all possible paths have been explored the end nodes are measured against the primary objective and the best one chosen.

In general, the space of alternative solutions looks more like a graph than a tree. Several paths (i.e., sequences of transformations) may lead to the same solution. Thus, it is important to detect points in the space that have already been examined. Other problems that arise in the exploration process have to do with the cost of the process itself. EXPL does not perform a brute force search. Accepting an alternative solution for further exploration depends upon the goals indicated by the user. Besides the main goals (speed, cost, and a trade/off factor) mentioned before, the user can also specify a minimum percentage gain for a transformation-derived solution to be acceptable. This is called the improvement factor. If the gain falls below the improvement factor, the new design is rejected.

This pruning process, when applied indiscriminately, can lead to an incomplete exploration. It may be the case that, although a derived solution is worse (according to the goals) than its parent solution, solutions derived from the former could in fact be better than the parent. EXPL handles the detection of this type of local optimality by allowing the user to specify a rejection level. The rejection level indicates whether or not non-improving solutions are to be further explored. The user specifies the maximum length of such non-improving paths.

The following section briefly presents several examples of design spaces. The examples illustrate some of the points discussed previously.

SAMPLE DESIGN SPACES

In this section we will present three examples of the design spaces explored by EXPL. We will not discuss the specific systems whose design spaces are depicted in Figures 14, 15, and 16. The examples will be used to show the characteristics of the design spaces and the exploration procedures.

Figure 14 shows the design space for a RTM system that is used as a controller for the X- and Y-plates.
of an oscilloscope. The system is used at CMU for RTM demonstrations (the “Munching Squares Generator”). Evaluations of individual designs are represented by a cross. A circle denotes that more than one design had the same evaluation. The first characteristic that can be noticed in this evaluation space is the stratification of the alternative designs. The solutions appear in horizontal bands representing solutions of similar cost. This is due to the high cost of the RTM buses compared with the cost of the other modules in the RTM set. The space is divided into bands corresponding to the 1, 2, 3, and 4 bus solutions.

The figure shows the degrading effect in RTMs of sharing variables between concurrent computations. The best solution (in terms of speed) used 3 buses and is faster than the 4 bus solutions. The algorithm is such that, although it allows a high degree of concurrency, when this degree exceeds a certain threshold there is a loss of speed in the total system due to the overhead of establishing the concurrency. The path followed to find the best solution is indicated in the figure. It is interesting to observe the transition from solution 2 to solution 3. There is a substantial gain in speed together with a reduction in cost. The explanation is that once the cost of a bus has been accepted as a reasonable price to pay for a given gain in speed it does not cost much to spread the load and perform more operations concurrently. Indeed, as the example shows, alternative allocation of the computations to the buses, for a fixed number of buses, is crucial.

When a solution is analyzed the set of feasible transformations that can be applied to its graph is tabulated. The improvement factor specified by the designer is then used to prune this table. This pruning takes place before a transformation is applied and is based on a preliminary “best case” analysis of a candidate transformation. The solution derived by applying the transformation may or may not realize the potential gain indicated by the preliminary analysis. This reduction in the predicted gain is due to several causes. If the goal is a reduction of cost, performing two concurrent operations in sequence may not in the case of RTMs result in a reduction in the number of buses (other computations may require the bus that was thought to be expendable). If the goal is a gain in speed, adding buses may result in a loss of speed due to the time required to copy and move variables between the buses in the system. Similar considerations can be applied to the case of Macromodules.

Figures 14 and 15 correspond to the design space for the same RTM system explored using the same cost/speed trade-off factor ($\$100/microsecond) but using different improvement factors. In the space shown in Figure 15, the preliminary improvement factor was set to a higher level (20 percent) than in the space shown in Figure 14 (10 percent). An interesting phenomenon occurred. The transformation indicated by the directed line in Figure 15 had a very promising preliminary evaluation (over 30 percent predicted gain). When the transformation was applied, the new solution did not realize the predicted gain. It was, nevertheless, better than the original solution and was later chosen by the system as the best solution. All feasible (i.e., applicable) transformations to this new solution were then examined and none of them promised to be better than the improvement factor. All of these transformations were rejected and the exploration path was terminated. When the same situation appeared in the example of Figure 14, there were several transformations that were better than the new, lower, improvement factor. One of them led in fact to the best solution of the space of Figure 14. It is interesting to observe in Figure 14 that the slope of the transformation from solution 1 to solution 2 indicates a better cost/speed trade-off than the transformation for the original solution (point 0) to solution 1. The gain in speed produced by the transformation, although smaller than the improvement factor used in Figure 15, was achieved completely; there was no overhead added to the system by the extra concurrency.

This type of anomaly is not uncommon in the modular design spaces explored so far, if anything, they tend to be the rule rather than the exception. The pruning of the applicable transformations, based on a preliminary analysis, can lead us to ignore certain
transformation paths that may yield better solutions. It is valid to ask, "why then should the system do any pruning at all?" The only reason we can provide is based on the analysis of the cases studied so far. Applying a transformation without any considerations to its possible gain is an expensive proposition. For any solution, branching factors (i.e., number of feasible transformations) of 30 to 50 are not uncommon. Applying a transformation implies a reconfiguration of the graph and the recomputation of several associated tables—an expensive operation in the current implementation. Applying each feasible transformation can lead to a very expensive design process. For example, when EXPL explored the design space of Figure 14, 454 transformations were analyzed. Of these, 337 were deleted for not meeting the improvement goal. The resulting 117 transformations (26 percent of the original) were applied to obtain new designs. Of these, 87 were deleted for not meeting the cost/speed trade-off or for being duplicated designs ("loops" in the design space exploration). As a result, only 30 transformations (6.5 percent of the original) produced designs that were improvements over their predecessors. For large systems, the combinatorial explosion of possible designs is prohibitively expensive. More intelligent heuristics are required to home in on the most interesting parts of the design space. Once these areas are located a near exhaustive search can perform the final selection. One such intelligent heuristic would be to identify the loop structure of the algorithm and apply transformations to the innermost loops. A variation of this might be transforming those loops whose (improvement factor) × (cycle count) product is largest. Furthermore, loop control and loop computation are parallelizable in many circumstances and their identification could speed up the transformation search which is seeking to identify any potential parallelism. Research into better heuristics is being actively pursued.

The system as implemented allows the designer to guide the exploration via an interactive command language. In the interactive mode, EXPL does not perform any pruning and the designer is free to order the system to perform any feasible transformation, regardless of its predicted gain. The automatic mode of exploration can therefore be used selectively under user guidance, assisting the application of the heuristics.

Figure 16 shows the design space for a system designed as a controller for a conveyor-bin unit. The design space corresponds to the alternative designs implemented using Macromodules. The figure is a good example of a design space with multiple paths leading to the same solution. The space configuration also indicates the characteristic of Macromodular systems. Once a basic design is implemented, variations in the level of concurrency do not present the radical changes in cost typical of RTM system. The basic costs of the Macromodular system are given by the memory and data operation modules (the "stack"). Variations in concurrency only imply adding or eliminating control modules and cables, a minor fraction of the total cost.

**VAS AND DESIGN VERIFICATION**

It is possible to develop an ISP description that is syntactically correct but that does not make sense logically. Figure 17.a depicts a syntactically correct ISP while Figure 17.b illustrates the corresponding graph. The graph is essentially the same one produced by the ISP compiler. The data operations have been deleted as a notational convenience (we can think of the data operations as being assimilated into the arcs connecting the control operations).

In the case of x = 1 the right half of the parallel merge in the graph would receive two control signals (one from the right half of the diverge, the other from the left half via the branch). The other input to the parallel merge would not receive a control signal and the system would deadlock at the parallel merge. Analytical tools based on the vector addition system (VAS) have been programmed to detect such design flaws.

The VAS is best introduced by example. Consider Figure 17.b. The arcs in the graph represent register transfers while the vertices represent control primitives. Each arc may contain tokens representing evolution of the associated register transfer. Graphically a token is represented by a dot on an arc. A marking of a graph with r arcs is a mapping from the set of r integers, each of which represents the number of tokens on the corresponding arc.

A vertex with a token on its single input arc is said to be enabled. Only enabled vertices can fire. The firing of a vertex removes a token from the input arc and deposits a token on its output arc. For the case of multiple input arcs there is an associated logic condition, either disjunctive (signified by a +) or conjunctive (*). A vertex with disjunctive input arcs is enabled when any input arc has a token. Firing the vertex removes a token from one input arc. This corresponds to a serial merge in the compiler producer graph. A conjunctive input condition requires tokens on all input arcs before the vertex is enabled (parallel merge). Firing the vertex removes a token from all the input arcs. Likewise a set of output arcs can be disjunctive or conjunctive. When a vertex with disjunctive output condition fires it places a token on one of the output arcs (branch). The conjunctive condition places a token on all the output arcs (diverge). A simulation is a sequence of permissible vertex firings.

Simulations are conveniently represented by the Vector Addition System (VAS). Figure 17.c depicts the VAS for the graph in Figure 17.b. The VAS con-
Properties of this tree can be used to detect properties of the graph. For example, the leaf \((0,0,0,1,0,0)\) represents a properly terminating sequence since there is a single token on arc 4. By contrast, leaf \((0,0,2,0,0,0)\) represents two tokens on arc 3. No tokens are on the exit arc. This is the deadlock situation alluded to earlier. Furthermore, depending on the actual physical implementation of the graph, this leaf may indicate a lost signal.

OTHER DESIGN TOOLS

One of the traditional design automation tools has been the simulation of a digital system. The RT-CAD system includes a simple simulator (SIMU10) which interfaces with the machine descriptions stored in the data base. Data base objects consist of two components: a symbol table and a flowchart. The former contains the description of the digital system component (registers, flags, memories, constants, etc.). The latter contains the data and control operations describing the behavior of the system. By stepping through the flowchart, SIMU10 can emulate the behavior of any digital system stored in the data base. An interactive command language allows the user to set and display the contents of the registers and memories of the system. The command language allows the user to set and reset arbitrary breakpoints and to interpret command files, created off-line, instead of prompting the user directly.

It is also desirable to produce designs according to criteria other than the traditional cost/speed criteria. One such criterion is testability. The structure of the final design substantially determines the ease with which tests can be generated for the design. A testability measure has been developed that correlates well with actual test generation effort. It is important to note that the common representation used as input to the various design programs is a critical feature that insures that the algorithm being evaluated is actually the one being implemented, verified, or simulated by the other design programs.

It is worth mentioning that some of the RT-CAD system components have been exported and are being used in places other than CMU. One of the more ambitious projects is the Computer Family Architecture (CFA) being developed by the Naval Research Laboratory. The objective is the implementation of a generalized Architectural Research Facility (ARF). NRL is currently using the ISP compiler and SIMU10. The ultimate goal is the implementation of a system that will emulate large machines described in ISP. Interactive facilities will assist the user in evaluating the described machine. Thus, proposed machines can be evaluated and their architecture tuned prior to production. The emulator will be entirely resident in the control storage of a dedicated PDP-11/40E (a commercially available, modified PDP-11/40 that
allows the coexistence of user written microcode together with the standard, DEC provided microcode that implements the PDP-11 instruction set).

FUTURE DIRECTIONS

To achieve the goal of automatic design relative to technology a mechanism is required that would take the description of a module set and create the equivalent of the ad hoc module set evaluators currently in use. It was also noted earlier that the order of physical allocation (registers, buses, operators, etc.) is a strong function of the design style imposed by the module set. This information would also have to be extracted from the module set description.

This preliminary design automation system and a machine relative optimizing compiler-compiler project serve as a stepping stone to an even more ambitious project termed the Symbolic Manipulation of Computer Descriptions (SMCD) depicted in Figure 19. There is a continual stream of new machines spurred by the advent of minicomputers and microprocessors. Each machine has a different instruction set. The emergence of microcoded systems with the option of user-defined instruction sets has increased this flow of instruction sets. Each new system requires supporting software and the amount of software grows for any individual system as user requirements grow.

One direction in which to seek a solution to ease the burden of software development is to relativize the production of software to the description of the machine. The central ingredient of this approach is the description of computer systems in a symbolic form, such that a range of problems can be solved by manipulation of these descriptions.

Figure 19 depicts the scope of the SMCD project. The ultimate goal would be to produce and evaluate a computer system from its behavioral specifications, together with the documentation and system programs. Thus the delay from the conception of a new architecture to the time it is implemented and ready for users can be significantly reduced.

ACKNOWLEDGMENTS

We would like to thank the many individuals whose work contributed to the results described in this paper. Steve Goldman, Ross Scroggs and Phil Karlton assisted in the development of the ISP compiler. Vittal Kini implemented the VAS, Don Thomas implemented the MM evaluator, and Steve Rodkey implemented SIMUI0.

REFERENCES

Some computer-related advancements for enhancing U. S. shipyard productivity

by RICHARD R. WISE and DOUGLAS J. MARTIN
IIT Research Institute
Chicago, Illinois

ABSTRACT

An exciting new cooperative program for enhancing productivity in the shipbuilding industry through improved automation technology is described. This program, designated as the REAPS (Research and Engineering for Automation and Productivity of Shipbuilding) Program, entails joint efforts of the Maritime Administration and the U.S. shipyards toward new capability development.

Four on-going hardware/software-related projects currently under way are described in some detail. These include: a major software system (AUTO-KON-71) for design and construction of ships; a specialized CAD/CAM remote batch graphics terminal to better serve shipyard needs; an interactive graphics system for pipe detailing; and a minicomputer-based numerically controlled frame bender.

A number of other hardware/software development programs currently approved but not as yet funded are also discussed.

THE NEED

Over the years a decided productivity lag has developed between U.S. shipbuilders and other shipyards around the world. For instance, a few years ago it was shown that U.S.-built container ships typically required 120 man-hours per delivered ton, while the equivalent ship built in West Germany required 90 man-hours per delivered ton. For ore carriers the U.S. required 75 man-hours per delivered ton whereas Japan required 59 man-hours per delivered ton.

Until 1970 the only recourse available to the U.S. Government to keep our Merchant Marine facilities competitive was to provide a subsidy program which reimbursed ship operators and shipbuilders the difference between the cost of materials and labor in the U.S. and those of competitive foreign countries.

In 1970, however, new laws were enacted which set in motion a large-scale research and development program in cooperation with U.S. shipbuilders and ship operators aimed at reducing costs and to improving productivity rather than merely reimbursing inefficiency.

THE SOLUTION

As a first step toward administering this program for the shipbuilding industry, the U.S. Maritime Administration convened a group of industry experts to identify where the most critical deficiencies existed and to provide guidance as to the best means for resolving them. The recommendations of this Technical Advisory Group resulted in the formulation of a number of MarAd supported research and development activities addressing six major development areas: computer automation, welding technology, surface preparations and coating, materials handling, ship producibility, and general projects.

In an effort toward developing a coordinated thrust in administering the resulting programs for applying automation and computer technology to shipbuilding, the REAPS Program was born. REAPS, an acronym for Research and Engineering for Automation and Productivity in Shipbuilding, is a cooperative program involving the U.S. shipyards and the Maritime Administration in a total system approach designed to identify and take advantage of productivity opportunities through the application of automation technology. Developments associated with this program are key to specific applications. Projects are not considered complete and successful until they have been implemented under shipyard production conditions. This implementation phase precludes the possibility of a project failing because of poor implementation of a sound development. It also insures that only those projects with valid objectives will be undertaken.

The REAPS program itself is funded jointly by MarAd and the participating shipyards and is administered by a contractor (IIT Research Institute). Its activities entail three major efforts: (1) Advance
planning to recognize future productivity opportunities; (2) Library and information services to apprise the industry of the latest available information on technology; and (3) R&D Program formulation.

The advance planning activity involves the identification of high-cost areas and subsequent target opportunities for new hardware and software research and development efforts.

The library and information services activity involves: (1) the quarterly publication of a Shipbuilding Technology Update Bulletin containing selected up-to-date citations and abstracts on world-wide publications of interest to shipbuilders; (2) maintaining a library of shipbuilding technology information and selected software programs; (3) distributing a library catalog and copies of the Update Bulletin to all major U.S. shipyards and providing a document reprint and software distribution service; and (4) holding an annual REAPS Technical Symposium where industry leaders are invited to discuss new advances in shipbuilding automation.

The R&D program formulation activity entails both project initiation and project monitoring. The advance planning activity identifies opportunities for productivity enhancements. Representatives from the REAPS yards then collectively prioritize these indicated requirements, develop project briefs defining detailed needs, solicit proposals from competent agencies for fulfilling these needs, and determine the best source for their realization.

The resulting development projects are then carried out on a cost-sharing basis with MarAd funds, with monitoring of all progress being pursued under the cognizance of the REAPS program.

ONGOING PROGRAMS

The remainder of this paper will be devoted to detailed discussions of the extant and contemplated computer-related projects within the REAPS Program.

AUTOKON SUPPORT PROGRAM

Background

The AUTOKON Support Program is actually the charter REAPS activity. Its genesis stemmed from actions taken by a U.S. shipbuilding industry advisory group in 1971 in response to a recognized void within the industry in the area of computer-aided shipbuilding. This requirement was most pressing for hull fairing functions and for numerically controlled burning of steel plates.

Accordingly, to resolve this deficiency, the Maritime Administration, acting on the advice of this Technical Advisory Group, purchased U.S. rights to a software system developed by a Norwegian Shipbuild-
fining a ship, in effect allowing it to be constructed within the database prior to its actual erection.

In addition to the usual economies in data maintenance and storage costs resulting from the minimization of data plethorism, further benefits are achieved in the design and construction cycle since all personnel using the system have access to identical information for consistency and efficiency.

With two exceptions (Modules ALKON and DUP—see below) each of the modules comprising the various AUTOKON-71 subsystems are applied in a simple straightforward manner involving submission of a minimal amount of data in fixed-field format on pre-printed data sheets. Following the commands and information on the input sheets, additional information, as required, is automatically retrieved from the database, appropriate computations are carried out, and results are recorded on the database and output listings, with graphical plots and/or N/C tapes also produced as necessary. The N/C tapes may be input to a drafting table, flame cutter, or alternatively, via a postprocessor, to a Tektronix CRT display or Calcomp drum plotter for quick-look review.

The two exceptions alluded to above are equally simple to apply but incorporate an interpretive command language in lieu of the fixed input format.

The entire system is structured to effectively fit within existing working conditions at a shipyard, with the underlying philosophy assuming that the users of the system are subject knowledgeable but have no background in computer processing.

Except for a handful of assembly-language programs for handling computer-dependent functions and inner-loop subroutines, the entire AUTOKON-71 system is written in Fortran. Thus, the system is almost completely computer-independent, with functionally identical versions operating on Univac 1100, IBM 360/370, Honeywell 6000, and CDC Cyber 70 systems.

A detailed discussion of the AUTOKON-71 database and each of its subsystems follows.

**Database management subsystem and associated utilities**

The database itself is physically assigned to direct access storage (drum, disc, etc.) and accumulates to the order of two million computer words for a typical ship, such as a large tanker or container ship. Data are stored in an arbitrary number of variable-sized indexed-sequential files each residing in one or more fixed-size physical blocks as required. Every file contains a number of variable-sized logical records (designated as “matrices”) each incorporating information of a specified type and having a specific format.

Files are addressed using a six-integer identifier. The meaning associated with each of these six integers is established according to a fixed set of conventions, which for each shipyard, are adapted to best adhere to local procedures and techniques.

The AUTOKON-71 database access and control subsystem is embedded in a module designated as the AUTOBASE module. These AUTOBASE routines, which are completely invisible to the user, provide an efficient access method for operating on the database. Whenever a specific six-integer identifier is referred for storage or retrieval (see Figure 2), a hashing algorithm is applied to the numbers to identify a unique page in the file catalog containing information on that identifier. That one page of the file catalog is then searched sequentially to find the most recent version of the desired file (grandfather versions of all files are optionally retained in the database to allow backup). The matching entry in the file catalog then contains a direct-access pointer to the desired file. The file, in turn, contains a File Table of Contents for accessing the desired matrix.

Thirteen independent utility modules are provided as an aid in administering the database. Nine of these are used to facilitate establishment and modification of the database, and three others for storing and retrieving data directly to and from the database, bypassing the other modules—useful for transferring data from N/C tape images, from a digitizer or to or from another database.

The last utility module, designated as DUP (Database Utility Program) is a stand-alone system for manipulating the database contents, for creating magnetic tape backup, for copying specified portions of the database onto another database, and for generating an annotated table of contents for the database. Unlike the other modules, DUP is an interpretive system using a set of commands specified free-form on punched cards. Typical commands include: READ, WRITE, DELETE, CONTENTS, COMPRESS, DUPLICATE, etc. Each command is usually followed by one or two parameters describing the data affected.

![Figure 2—Database access mechanism](image-url)
Hull fairing subsystem

The AUTOKON-71 hull fairing subsystem is actually comprised of three modules: FAIR for the actual fairing of the ship's lines; DRAW to generate N/C drafting table drawings of the faired lines; and TRABO for TRANSferring the final BOdy plan to the AUTOKON database. As shown in Figure 3, the fairing process is carried out using a sequential file for storing the offset information. Because the fairing activity involves an iterative man-machine process, significant economies can be effected by performing the initial fairing runs independent of the database, incorporating only the final faired lines into the database.

The purpose of the FAIR module is to perform the fairing of ship lines and to transform these lines into a numerical form suitable for further computer processing. The basis for performing the fairing process is a preliminary lines drawing in a suitable scale. Normally 200-400 data points for an entire ship are lifted from the lines drawings, the number depending upon the extent of shape incorporated in the hull form.

Input consists of some key curves and selected points on frames, waterlines and/or buttocks. The fairing is performed automatically from the data points selected by the user, and results are presented in printed form and, via the DRAW module, in N/C form for subsequent drawing on an N/C drafting table. Drawings are automatically produced, as desired, for all curves faired.

In use, FAIR may be employed as desired to generate an entire ship or as little as one-fourth of a ship. The different parts of the ship may then be "hooked up" to form the complete ship. As any of these hull representations is satisfactorily developed, it may then be transferred from the working storage used by FAIR into the database for the desired hull and added to whatever previously exists. This body plan transfer is accommodated through the use of the TRABO module.

Longitudinal detailing subsystem

Once the hull data are incorporated within the database, this subsystem can then be applied to strake the shell by fitting longitudinal curves to the hull surface. Using a module called LANSKI (a Norwegian mnemonic standing for longitudinal details), contours for seams, longitudinals, stringers, bulkheads, decks, etc. can be automatically defined.

Input to the program consists of a few points for each seam and for each longitudinal from its start point and end point, together with a simple identification of desired cutouts and of the first and last transverse frames along the shell to which the cutouts are to be applied (also using INCLUDE or EXCLUDE operators to identify explicit frames where cutouts are to be omitted).

As output, a Table of Details is produced, both in the database and as a tabulated listing, describing how and where the longitudinals, seams, and decks cut through transverse frames, bulkheads, etc. The space angle between two intersecting parts is computed and stored to provide automatic correction of the nominal dimensions for clearance, on height, width, and, if necessary, on the thickness and angle of flange, etc. Plots of the body plan with the longitudinal curves superimposed, and other drawings for checking purposes are produced.

Parts programming subsystem

The subsystem for parts detailing, designated as ALKON (standing for ALgorithmic CONstruction), is by far the most sophisticated element of the AUTOKON-71 system. Input for this module is specified in a very powerful Fortran-like problem-oriented language. While it is primarily a plane geometry definition tool, ALKON boasts of many other powerful features including:

- Fairing of individual curves
- Detailing of complex steel structures
- Production of drawings with augmented text (drafting automation)
- Production of data for N/C control (drafting, flame cutting, milling, etc.)
- Definition and printing of miscellaneous tabular information
- Definition of standard details (parametric descriptions of stringers, scallops, etc.)
- Formalization of design and production practice (accumulation of experience)
- Definition of identification system and data structures
The ALKON language is completely flexible, providing the user with tools to replicate the functions of literally any of the other AUTOKON modules if desired. It should be noted that while the grammar and meaning for every vocabulary word is explicitly defined within the system, the spelling associated with each of these words is not; thus each yard is able to define its own version of the vocabulary, incorporating whatever abbreviations and localized spellings are desired, taking full advantage of local customs and nuances.

One of the most powerful adjuncts of the ALKON language is represented in its ability to allow definition of norms (i.e., subroutines and functions). Norms may be given dynamic names and may be defined and called at any time. Norms are stored in the database just as any other AUTOKON information, are re-entrant, and may be executed recursively.

Since norms are, in effect, extensions of the ALKON language, the capabilities of the ALKON system are virtually limitless. In practical use within a shipyard, norms are applied in a hierarchical manner: Very basic norms are used to define elementary items such as cutouts, holes, etc. Other norms combine them into series of holes, contours with cutouts incorporated, etc. Still other norms can be used to generate complete parts to produce web frames, floors, etc. It is even theoretically possible to write a norm to design an entire ship.

The ALKON processor itself is organized in two passes, translation and processing, which are executed in sequence. All input for a run is translated before the processing pass is activated, with all data transfers between the passes being handled by the AUTOBASE routines.

Input to ALKON is normally on punched cards. This input information is read into the ALKON processor one card at a time by the Translation module. Each line is listed on the line printer and is then translated from the higher-level, external representation of words and numbers into a binary format. At the same time, a few special symbols are interpreted with appropriate actions taken, and a check is made on the formal correctness (syntax) of all the other input code. Misspelled words are rejected, missing parentheses are noted, etc. If complete fields in a statement are missing, a default value is substituted. This default feature makes it possible to write statements in a very simple way for common situations while allowing the same statements to be used in a sophisticated manner for more complex situations. Vocabulary definitions are also accommodated at this point as well as Comment statements which are recognized and immediately discarded.

All the major fixed internal logic for Translator processing is handled through the use of three sets of joblists together with vocabulary tables and decision tables. Through the use of these tables, all other statement types not mentioned above are efficiently translated from the character representation of the input to a binary (inverse Polish) notation and passed on to the Processing Module via a set of database records designated as code blocks. Newly defined norms are also retained as code blocks as are previously-defined norms.

Following completion of translation, control is then passed on to the Processor. Here the code blocks for manuscripts are returned from the database one at a time, and the statements are acted upon. If a call for a norm is encountered, the processing of the manuscript is temporarily suspended while the statements of the called code block are acted upon. If another norm call is encountered, the processing of the present code block is again suspended and the new code block acted upon. This process may be repeated to any depth, the processing of the previous code block being resumed when the processing of the called code block is completed.

When processing of the code block for a manuscript is finished, the next manuscript code block is then retrieved from the database and processed in the same manner. Finally, when there are no more manuscript code blocks, processing is completed and the present ALKON run is terminated.

The ALKON code for a simple part is shown in Figure 4. As previously indicated, the fairing subsystem results in the storage of the ship’s lines in the database. The use of simple norms for retrieving this information is illustrated in Figure 5. The norm calls

\[
\text{FETCH TFRAME (89) AT SHELL'}
\]

instruct the system to retrieve the file containing the contour defined by the intersection of Transverse Frame 89 with the outer shell of the ship. The norm calls

\[
\text{FETCH LICON}'
\]

then instructs the system to retrieve the specific matrix in the file for the desired lofting contour. This code can then be embedded within a new contour definition, as shown in Figure 6, to define a part made up of this contour.

The Longitudinal Detailing subsystem generates tables of details for all required cutouts and stores them in the database. The ALKON code shown in Figure 7 illustrates how the wire frame norms can then be applied to automatically modify a lofting contour to incorporate these cutouts. The norm

\[
\text{GEN ACON ( )}
\]

retrieves the various tables of details, previously stored by the LANSKI module, references the appropriate norms for the cutout definitions and automatically generates the augmented contour incorporating all the desired cutouts.

*Plate development subsystem*

Concurrent with the parts detailling activity but somewhat after the LANSKI runs incorporating the
**ALKON MANUSCRIPT**

```
TEMPS
STRT LGEO
ON (CT)
SPT
SL: EPT(1000)
SL: EPT(1000 + 600)
SL: DIR(180) EPT(200 + 600)
CIR: SDIR(180) EPT(0 + 400)
SL: EPT
END LGEO
NC CON

(0,000,000)
```

**Figure 4—ALKON code for simple part**

![Figure 4](image)

**ALGON MANUSCRIPT**

```
TEMPS
STRT LGEO
ON (CT)
SPT
SL: EPT(1000)
SL: EPT(1000 + 600)
SL: DIR(180) EPT(200 + 600)
CIR: SDIR(180) EPT(0 + 400)
SL: EPT
END LGEO
NC CON

(0,000,000)
```

**Figure 5—Usage of wire frame norms**

![Figure 5](image)

**Figure 6—ALKON code for web frame**

```
TEMPS
TFRAME (89) AT SHELL' FETCH LCON'
STRT RGEO'
ON (CT) SPT (<3> + <33,4,9>)
SL: DIR(0) INT(<47> + <33>)
CON' INT'
SL: DIR(90)
EPT (<3> + <33,4,9>)
END RGEO'
```

**Figure 6—ALKON code for web frame**

![Figure 6](image)

**LOFTING CONTOUR—FRAME 89**

```
NEW TBUF'
TFRAME (89) AT SHELL'
GEN ACON (TBUF + 1)
```

**Figure 7—Augmenting lofting contour with cutouts**

![Figure 7](image)
seams and butts in the database, use of the shell plate development subsystem normally commences. This subsystem is comprised of two AUTOKON-71 modules, aptly designated as SHELL and TEMPLATE.

Shell is used to develop the steel plate required for the outer shell of the ship and TEMPLATE generates the roll sets to be used in forming them. For each desired shell plate, input consists of butt locations (stated as axial distances from transverse frames) and seam identifications together with an indication of which of two alternate expansion methods is desired. The SHELL program retrieves the required hull form and longitudinal curve information from the database and generates 2-axis or 3-axis N/C tapes for cutting and marking the plates. It also generates N/C drawings in various scales to be used for checking, planning, or optical burning. The locations of the rolling lines and any stiffeners are automatically punch marked on the plate, or indicated in the drawing by dashed lines.

The TEMPLATE program uses essentially the same input and generates N/C tapes for plots of roll sets for the shell plates, or the roll sets themselves. Information for three templates is produced for each plate: at the first transverse frame, the last frame and the middle frame. This module may also be used to generate dimensions for the production of 1:1 wooden templates for shaping transverse frames.

Part nesting subsystem

Most parts developed with the parts programming subsystem are not directly cut but rather are stored on the database for subsequent nesting on a rectangular steel plate so that the steel is used efficiently. The AUTOKON-71 NEST module is utilized for this activity. Working with appropriately scaled part drawings, the placement of the parts on the raw plate is determined manually. Input to the NEST module then consists of the part numbers for each of the nested parts, their relative location and orientation (normal or mirror) on the plate, approximate cutting bridge placement, and (forward/backward) cutting sequence for each part. The program automatically retrieves the contour and auxiliary function information from the database for each part specified, and generates the required N/C tape for plotting and flame cutting of the nested format. The plots can be used for checking, planning or optical cutting.

Planning and production data subsystem

This subsystem is embodied in the AUTOKON-71 PRDO (PROduction Data) module which is used to extract data from the database to provide useful information for planning and production purposes. Output produced includes: (1) burning time and cutting length for nested formats and single parts; and (2) area, volume, and weight information for single parts or, optionally for an aggregate series of parts.

Design analysis subsystem

This subsystem, as incorporated within the AUTOKON-71 system, is collectively designated as the PRELIKON (Norwegian mnemonic for Preliminary Lines) package. It is comprised of 23 modules which are used for both preliminary and final naval architecture calculations and for automatic production of various tables and plots normally required on delivery of the ship.

All computations are performed on an independent database. Initial establishment of this database is achieved with the Hull Definition module or the Hull Variation module, the latter of which can be used to apply minor changes (LCB, CB, or main dimensions) to elements of a library of hull forms.

The AUTOKON-PRELIKON linkage module, which, in effect, retrieves the body plan from the AUTOKON database and maps it into the PRELIKON database, may be used later to allow the modules below to be applied for final calculations.

Modules are also provided for the following computations:

- Hydrostatics
- Load Distribution and Balancing
- Resistance
- Bon Jean Tables and Plots
- Transverse Stability
- Floodable Lengths
- Launching
- Trim Tables and Plots
- Capacity, Sounding, and Ullage Tables
- Grain Stability
- Longitudinal Strength

Program support activities

The AUTOKON Support Program can be defined as a special interest activity being pursued under the auspices of the REAPS Program. As previously stated, the basic purpose for this program is to provide an organized mechanism for maintaining and enhancing the AUTOKON-71 system for optimal use in the U.S. shipbuilding environment. Activity towards this end falls into three categories: Software Support, Information Dissemination, and System Enhancements Requirements and Planning.

For software support, all the expected system support activities are being pursued under this effort. A formal mechanism has been established for reporting system failure information for subsequent resolution by the REAPS Program staff. Periodically, all the accepted fixes are incorporated into the system and a
new Standard System is distributed to the participating yards. Tapes and complete update documentation are provided. Implementation assistance is also made available to new system users. A new four-volume AUTOKON Users Manual has been written and made available to participating yards, and training courses have been developed and given.

Proposals for major system capability enhancements are periodically reviewed by the AUTOKON Support Program Technical Representatives, with resultant recommendations made for MarAd funding for those deemed to merit implementation.

PIPE DETAILING SYSTEM

The following describes the results of a preliminary systems design recently carried out by Newport News Shipbuilding and Dry Dock Co. in the first phase of a two-phase effort to design and implement a minimum cost system for pipe detailing. The purpose of this effort was to identify an input station configuration for use in digitizing piping design data that would substantially reduce the cost of preparing input data for various computer-based systems used for the production of pipe manufacturing documents.

Operational overview

Functionally, the desired system must accommodate four major tasks: data input, document production, data revision and updating, and data communication. These tasks are described below in the context of a typical operational sequence.

At the beginning of a typical design session, the designer will retrieve the appropriate piping arrangement plan or preliminary design drawing and mount this document on an X-Y digitizer, along with the command menu list to be used in the input process. Next he must retrieve from a hull structural database (e.g., an AUTOKON database) the necessary structural reference data (transverse frame, deck, bulkhead locations, etc.) such that geometric data for the piping system may be conveniently referenced to these locations. Using these reference locations, the designer will then calibrate the drawing(s) on the digitizer by touching the positions of these structural landmarks with the free cursor. The system is then prepared to accept digitizer input.

The designer then proceeds to input the piping system geometry and details utilizing the digitizer, a keyboard or CRT cursor picks. These picks relate to a CRT display of the current definition of the piping system which is built up during the input process. The CRT display also serves as a check on the designer's work as he proceeds.

Keyboard input of piping geometry may be in the form of coordinates relative to the structural reference locations retrieved from the hull database, absolute coordinates or coordinates relative to other points already defined in the pipe geometry itself.

An important feature of the system is the capability to automatically select specific components based on a predefined set of component selection rules applicable to a particular piping system. This feature, for example, will allow the designer to specify a condition generically (e.g., TURN) during the input process. Then, a post-processing operation will automatically determine the specific fitting, etc., to be substituted (e.g., a bend or a 90° or 45° elbow) for the generic condition through an analysis of the component selection rules.

Regardless of whether components are explicitly referenced by the designer, or automatically selected as described in the preceding, the system will proceed to automatically determine the required orientation in space of the component based on its local geometry and the geometry of the connected piping. Additionally, error checks are performed on the geometry of bent pipes, to insure that they may be produced by the yard's pipe bending machines, as well as on joints in the piping system to verify the compatibility of members at the joint in terms of diameter, end type, and alignment.

At the completion of an input session, or at any time during the input process, the designer may request the system to produce checking documents. These include a centerline check print (plot), consisting of a single line drawing of piping centerlines drawn to the same scale as the drawing from which the piping geometry was lifted such that it may be overlaid on the original, a piping system data list, and a listing of missing data (see Figure 8). This latter list is required if the designer is working from a preliminary design drawing which will not contain all necessary detail information required to complete the design. In this case, the system notes the locations where information is lacking, but allows the designer to proceed with the input process. Then, at a later date, when the designer has determined which details are to be used for this specific system, he can return to the incomplete system definition to fill in the missing information.

Figure 8 is an example of such a checking document as might be produced by the system. Note that graphical prototypes of the defined fittings are also presented. Also indicated are two sectional views of the system, defined by the user. The user may select views of the system by one of three methods:

1. by identifying several structural reference points which are to be included in a plan, section or elevation view;
2. by identifying a pipe leg which will serve as a view axis or a pair of intersecting legs the normal to which will serve as the view axis;
3. by specifying cross-sections through an existing view (e.g., Figure 8).
At the conclusion of an input or modification session, the designer may either dump the specific piping system database to magnetic tape for later retrieval and additional detailing work or, if the system is complete, transmit the database to the central database for accessing by engineering analysis, material and production control, and manufacturing document production programs.

System configuration

Figure 9 is a schematic representation of the proposed input terminal configuration. Each such terminal may support from one to four input stations each consisting of a digitizer, a keyboard, an output graphical display device and an output alphanumeric display device (either a teleprinter or character CRT). The box labeled “Printer” may actually consist of one of two alternative configurations; an incremental pen plotter for the production of centerline check prints and a medium speed line or character printer for producing data lists, or a single electrostatic printer/plotter which would satisfy the requirements for both applications. The communications interface and modem will support 1200 baud synchronous communications between the minicomputer and the central site facility, a Honeywell 6080, via a Datanet 355 communications front-end.

All database operations will be implemented within the framework of a vendor-supplied database management program which conforms to the recommendations of the CODASYL committee. A database network structure has been defined for the representation of piping data interrelationships which uses a subset of the CODASYL specifications and which is compatible with several commercially available minicomputer-based DBMS.

The final, or development phase of the work reported here is currently under way at Newport News and is scheduled for completion in early 1978.

SHIPYARD GRAPHICS AND COMMUNICATIONS TERMINAL

The following discussion reports on the results of a design effort carried out by IIT Research Institute to develop a low cost remote terminal configuration for use by shipyard loft departments in processing the input and output of N/C software systems such as AUTOKON and others.

Background

This project was initiated with the intent of developing a terminal configuration which was modular, for ease in configuring various throughput/cost configuration alternatives, as well as to minimize the cost of each such configuration throughout the range of these capability alternatives.
Functional requirements

The functional requirements of the terminal, developed as a result of interviews with the personnel of various shipyards in the U.S., are as follows:

1. The terminal must provide a Remote Job Entry (RJE) capability for a variety of mainframes via emulation of various remote batch terminal protocols typically associated with these mainframes.
2. It must provide a local facility for reviewing numerical control (N/C) data graphically on a cathode ray tube (CRT) display and on an automatic drafting machine (ADM) in the following modes:
   a) ESSI verification
   b) Drafting
   Additionally, the system must be capable of punching, on paper tape, the final verified ESSI file for transmittal to the burning shop to be used to direct N/C flame cutters.
3. It must provide the necessary data management and storage facilities to support the above-mentioned activities efficiently.
4. It must finally, provide an operating environment in which software tasks associated with the above functions may execute concurrently in servicing one or more users.

Operational environment

The terminal user/operator, as with other remote batch operations, will load the appropriate RJE emulation software into memory from local mass storage and establish communications with the central site facility. Once loaded, the emulator is prepared to accept for transmission a user input deck. The user loads the deck in the card reader and instructs the emulator to transmit the input stream to the central site.

At the central site, the appropriate N/C software system module is invoked and executed and the output of the run queued for transmission back to the originating terminal site. Output from these programs is of two forms: the normal printed output (or Print File) which will be directed to the terminal's line printer, and the resultant ESSI code describing the geometry of the part, plan, etc., produced by the run. This file will be treated by the central site as a Punch File and addressed to the terminals' punching device. In practice, however, this file will be automatically written to the terminal's local mass storage device and catalogued under a file name consisting of the unique run identification information assigned by the central site.

Upon reception of these files, the user may then referring to the run I.D. appearing on this printed output, initiate a graphics session at the CRT console and call out the appropriate "Punch File" for graphics processing.

The graphics processing system will allow the user to plot the ESSI data on the CRT for a quick check of the data or produce an ADM drawing of the data by simply specifying the desired output device. If the user of the CRT is not the originator of the run, this terminal operator may simply produce a hardcopy of the CRT check plot on the optional hardcopy device for delivery, along with the printed output listing, to the run's originator for review. After review of the run results, the loftsman or designer who initiated the run may request the operator to either plot the catalogued Punch File on the ADM, if the results are satisfactory, or purge the file if an additional run will be required. A punch tape of the file will also be requested if the file is to be used by the N/C burning shop.

System description

Figure 10 represents a typical configuration of the terminal being described. One of the more interesting features of the system is the block labeled "Table Control." This is a microprocessor-based control unit containing all the required logic for linear and circular arc interpolation and the generation of slope and acceleration control required by the head drive electronics of the ADM itself. This capability relieves the system processor of the requirement of computing and outputting incremental positioning data for each step of the ADM's resolution (i.e., 0.001').

The controller can be interfaced to the system processor via standard EIA serial interfacing, further simplifying the overall terminal configuration.

The CRT device is of the storage tube variety and is also interfaced to the system via an asynchronous EIA RS 232 interface. It will operate nominally at 9600 baud. It is equipped with a thumbwheel cursor for use in the graphics processing software for digital coordinate input.
The system console may alternatively be a character CRT display or teleprinter device.

Graphics software

As mentioned previously, two distinct modes of operation of the graphics processing system are available. The first, or ESSI verification, mode of operation in addition to producing plots of geometry described by the ESSI data, also checks the ESSI auxiliary function codes (e.g., center torch on/off, rapid traverse on/off, left/right kerf width compensation on/off, etc.) for consistency. In this mode, auxiliary function code annotation is added to the geometry display/plot and errors reported (e.g., both center torch and rapid traverse on concurrently) on the user console.

The second, or drafting mode, ignores all but the center torch auxiliary functions which serve as the pen up/down commands. This mode is of particular value in reviewing ESSI output from the N/C software modules which are not directly concerned with flame cutting. These would include, for example, the AUTOKON FAIR and LANSKI (see the discussion of AUTOKON in this paper) modules which output ESSI representations of hull lines (e.g., bodyplan, buttocks, waterlines, shell plate seams, etc.) and longitudinal structure traces. For these applications, the user may make use of the graphics processor's windowing (or zoom) capabilities to good advantage to examine more closely potential problem areas requiring greater visual definition. Also implemented in the graphics system software are 3-dimensional display and transformation routines to support future applications in the design area.

Capabilities summary

A modular, multi-user RJE/Graphics terminal system for shipyard use has been designed. The system’s capabilities (summarized in Figure 11) include RJE communications, for a variety of main-frames, which may be executed concurrently with one or more graphics processing sessions or with various utility programs operating in background mode. The graphics processing software allows interactive view modification at a CRT or a simple batch type mode of operation applicable to both CRT and ADM.

N/C FRAME BENDING MACHINE

The following describes a development effort carried out by Case Western Reserve University which has resulted in the production of a prototype, fully automated frame bending machine.

### TABLE I—FRAME BENDER CONTROL ALGORITHM

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>OPERATOR INSERTS BEAM INTO FRAME BENDER</td>
</tr>
<tr>
<td>2.</td>
<td>OPERATOR CLAMPS TRANSDUCER AT END OF BEAM</td>
</tr>
<tr>
<td>3.</td>
<td>INPUT THE DESIRED BEAM SHAPE</td>
</tr>
<tr>
<td>A.</td>
<td>ESSI model of circular arcs and straight line segments (can be piecewise linear)</td>
</tr>
<tr>
<td>B.</td>
<td>From AUTOKON paper tape, teletypewriter, or disk file</td>
</tr>
<tr>
<td>C.</td>
<td>ASCII or EIA character codes</td>
</tr>
<tr>
<td>4.</td>
<td>INPUT THE BEND PARAMETERS</td>
</tr>
<tr>
<td>A.</td>
<td>Work length minimum and maximum</td>
</tr>
<tr>
<td>B.</td>
<td>Initial unbent length at end of beam</td>
</tr>
<tr>
<td>C.</td>
<td>Clamping mode</td>
</tr>
<tr>
<td>D.</td>
<td>Tolerances for feed distances and bend angles</td>
</tr>
<tr>
<td>5.</td>
<td>CALIBRATE TRANSDUCERS WITH A/D CONVERTERS</td>
</tr>
<tr>
<td>6.</td>
<td>SET UP IDEAL MATHEMATICAL MODEL OF BEAM</td>
</tr>
<tr>
<td>A.</td>
<td>Work points are preferred at junctions of ESSI elements</td>
</tr>
<tr>
<td>B.</td>
<td>The last work section may overlap the second last one</td>
</tr>
<tr>
<td>C.</td>
<td>The tangent vector at each work point is determined from the ESSI model</td>
</tr>
<tr>
<td>D.</td>
<td>A table summarizing ideal model of beam can be printed</td>
</tr>
<tr>
<td>7.</td>
<td>EACH WORK SECTION IS PROCESSED AS FOLLOWS:</td>
</tr>
<tr>
<td>A.</td>
<td>Feed beam and adjust moving head to position new work section between the fixed and moving heads</td>
</tr>
<tr>
<td>B.</td>
<td>Update model of actual beam to reflect feeding</td>
</tr>
<tr>
<td>C.</td>
<td>If necessary, move transducer to new point on beam, find new reference point on model of actual beam, and find corresponding point on ideal model</td>
</tr>
<tr>
<td>D.</td>
<td>Calculate (X, Y) aim coordinates for transducer reference point from ideal model</td>
</tr>
<tr>
<td>E.</td>
<td>Bend to Y coordinate of aim point, release, and measure springback</td>
</tr>
<tr>
<td>F.</td>
<td>Until bend angle tolerance is satisfied (but never more than 2 iterations), recalculate required “overbend” based on springback just observed, bend to new Y coordinate, release, and measure springback</td>
</tr>
<tr>
<td>G.</td>
<td>Update model of actual beam to reflect bending of this work section</td>
</tr>
<tr>
<td>H.</td>
<td>Feed to X coordinate of original aim point</td>
</tr>
<tr>
<td>8.</td>
<td>WHEN ENTIRE BEAM IS FINISHED, OPTIONALLY PRINT A TABLE SUMMARIZING THE MODEL OF THE ACTUAL BEAM</td>
</tr>
<tr>
<td>9.</td>
<td>OPTIONALLY PRINT A TABLE COMPARING THE IDEAL MODEL WITH THE MODEL OF THE ACTUAL BEAM</td>
</tr>
</tbody>
</table>

### Background

Historically, frame bending in the shipyard has been carried out by one of two methods: hot slabbng, which requires the shape to be furnace until the material becomes plastic and then forced against a full-sized template through use of a hydraulic ram, or three-point cold form bending, in which the operator of the three-point machine iteratively applies a force on the shape between two fixed support points until the desired curvature has been produced.

Each of these techniques has substantial drawbacks. Both require the production of full-sized templates. The hot slabbng process requires a furnace facility...
CAPABILITY SUMMARY

1. TASK CONCURRENCY:
   - RJE Communications with Graphics Monitor Job
   - OR -
   - Background File Manipulation
     (e.g., Disk File to PTP
     PTP to Disk File
     Disk to Disk Copy
     Text Edit)
   - Program Development, etc.

2. ESSI Drafting Plus Verification For Either CRT or Drafting Table
   Features:
   - Windowing
   - Zoom
   - Rotation
   - Mirror Image
   - Auxiliary Function Annotation
   - Drafting Table Control
   - Console Emulation
   - 3-D Capability for Future Applications

3. EXPANDABILITY
   - To 128K Words Memory
   - To 8 Disk Drives
   - Multiuser
   - Multitable

Figure 11—Shipyard graphics terminal capabilities summary

and is very labor intensive. And the cold bending is slow, requires a skilled machine operator and, most significantly, produces out-of-plane bending, buckling and twisting (i.e., the plane of deformation of the shape and the plane in which the bending moment is applied may not coincide).

The CWRU frame bender represents a unique solution to these problems. The prototype itself is a 1/6 scale version of a full scale device which can accommodate interchangeable dies, automatic clamping, automatic feeding and provision for operation in either a completely autonomous, computer-controlled mode or by manual control.

Mode of operation

Figure 12, reproduced from Reference 2, depicts the control loop involved in directing the machine. Feedback signals from the transducer group are fed through a multiplexed 12-bit A/D converter to the computer (a 32k Nova 2/10). Based on the current and desired frame contour (acquired from the ship structural database, e.g., AUTOKON) the computer calculates the necessary bending to be applied and outputs this data, via a D/A converter, to control hydraulic valves on the machine. This process is an iterative one in which short segments of the frame, called work sections, are processed sequentially. For each work section a target or “aim” point is computed such that cumulative error is minimized. The work section is then bent, taking into consideration the approximate springback of the frame, in an attempt to position a reference point near the original end of the frame on the aim point. If the two points do not coincide after bending and springback, the work section is again bent on the basis of the springback measured after the last bending operation and the deviation of the reference point from the aim point. Throughout this process, for frames with non-symmetrical cross-section, hard wired logic maintains the appropriate in-plane bending through control of a servo valve.

Three separate mathematical representations of the frame are maintained by the minicomputer during the bending process. The first is the ESSI (i.e., N/C contour) model of the frame as retrieved from the AUTOKON database. The “ideal” model is derived from this ESSI model prior to the initiation of bending by segmenting the ESSI elements into work sections (see Figure 13) according to a set of segmentation rules, and computing vectors tangent to the ESSI model data at all work points for alignment of the frame. Finally, the actual model, developed during the bending process, represents the true shape of the frame based on actual curvatures applied and work section lengths fed. From this model, a table of discrepancies can be produced relating the final actual to desired shape.
FUTURE PROGRAMS

Consistent with a broadening of the technical scope of, and shipyard participation in the REAPS Program, is a shift in emphasis in the program's R&D efforts to the development of programs or systems which do not rely on any of the several individual N/C software or other production-oriented systems for applicability. Some examples of projects to be undertaken in the near future are listed below.

1. Structural Detailing System—for definition of stiffener intersection details including N/C descriptions of end-cuts for subsequent use by N/C fabrication equipment.
2. A Low Cost Parts Definition System—for quickly entering or modifying existing part geometries in an interactive mode through use of a minicomputer-based digitizer system.
4. Structural Assembly Aids System—for producing parts explosion type drawings of structural units to assist assembly crews in fabricating structural units quickly and accurately.

Such developments it is felt will now and in the future provide U.S. shipbuilders with cost-effective technological problem solutions which will serve to enhance shipbuilding productivity.

REFERENCES

Computer analysis and evaluation of marine structures

by DONALD LIU and MATIAS E. WOJNAROWSKI
American Bureau of Shipping
New York, New York

ABSTRACT

The marine industry in general and the American Bureau of Shipping in particular have turned to the extensive use of computers for the solution of the problems encountered in the design, analysis and construction of ships and other marine structures. Mathematical solution techniques have been available for some time, but the complexity of the necessary calculations precluded their use. The advent of electronic digital computers with their powerful, constantly expanding "number-crunching" capabilities has bridged this gap, leading to the application and further expansion of available techniques.

The most useful and usable tool is the finite element method, which has found widespread use and has been extensively applied for structural, dynamic and thermal analyses of marine structures, ranging from entire vessels to local structural details. Descriptions and examples of such analyses are given in the paper.

Other facets of computer versatility applicable to and used by the industry include conversational programs, naval architecture programs, computer graphics and information retrieval. These subjects are presented and discussed in the paper.

Looking ahead, future trends of computer applications in the industry are mentioned.

INTRODUCTION

The American Bureau of Shipping (ABS) is a ship classification society which performs the primary service of certifying the soundness and seaworthiness of merchant ships and other marine engineering structures. ABS establishes standards known as "Rules" for the design, construction, and periodic survey of vessels. By applying these internationally accepted "Rules", ABS classes ships, that is, assures that ships are fit for their intended service. Classification provides assurance to owners, purchasers, shippers, underwriters, and other interested parties that a particular vessel possesses the structural and mechanical capability for safe performance.

Realizing the power of the computer and its application as a valuable engineering tool for vessel classification services and such fields as naval architecture and structural engineering, the American Bureau of Shipping has been in the forefront of the development and usage of computer programs in the marine industry. As a service organization to the industry, ABS has also adopted a policy of making known to interested parties the methods it employs in computer-related activities. This paper is a small contribution toward that goal.

The complexity of engineering problems encountered in the marine industry has led to extensive and ever-expanding computer usage. In addition to the conventional static and dynamic problems of design, the problems encountered in assessing marine structural response are compounded by the unpredictable nature of sea, and sometimes cargo, loads. Sea conditions represented by wave heights, wave lengths and wind speeds are measured and the information is compiled statistically. Structural design is based on sea conditions with a probability of occurrence of $10^{-4}$, which corresponds to an event occurring once during an assumed ship service life of approximately 25 years (20 years at sea). Special phenomena associated with the dynamic interaction of waves and ships at sea must be taken into account. Springing, for example, is a vibration of the complete vessel induced by the wave frequency in conjunction with the ship's elastic properties. Other areas of concern are local vibrations, which may be induced by waves or action of the propeller and drive shafts. Other loading conditions include those due to thermal effects, sloshing of liquid in cargo tanks and sea ice.

Mathematical techniques, such as matrix methods, finite element methods and statistics, have been available for a long time. It has taken the advent of electronic digital computers to make possible the full utilization and implementation of these techniques into an efficient engineering approach to the solution of the numerous problems associated with the design, construction and analysis of ships and other marine structures.
FINITE ELEMENT METHODS

The finite element technique is a relatively new and very useful method for stress analysis of structural continua. The method relies strongly on the matrix formulation of structural analysis introduced mainly as a result of the increasing use of computers. There has been a concurrent and rapid development of electronic computers, matrix methods and finite element techniques.

In the finite element method, a solid continuum is subdivided into an assemblage of discrete elements of finite dimensions. In effect, the real system or structure is modeled basic a simplified, idealized system for which a solution is available. The idealized model is then analyzed by one or more of the available methods of analysis.

The finite element technique has a sound theoretical basis within the framework of the classical theory. It may be interpreted as a close relative to the well-known Ritz method, in which the displacement field in a continuum is usually described by means of a sum of pre-selected functions, each multiplied by a constant. The constants are determined by means of the condition of minimum potential energy.

While in the classical Ritz procedure one set of functions describes the displacement field in the entire continuum, the finite element method assumes individual displacement fields for each of the elements. The internal displacements in the elements are uniquely defined in terms of the nodal point values, and the entire displacement field is assumed to consist of a large number of piecewise continuous fields, each covering the extent of one element. The conditions of equilibrium of the nodes may be shown to yield the displacement field corresponding to minimum potential energy for the selected displacement pattern. As in the Ritz procedure, the solution will generally be approximate, but the method converges toward the correct solution.

Many new developments and refinements are constantly being added to the existing finite element programs. One of the most promising of these extensions, extremely useful in the analysis of complex structures, is the substructuring capability. Substructures are assemblages of basic elements (beams, plates, etc.) which serve as building blocks for the representation of larger, more complex structures. Substructuring decreases the amount of input data required to generate the structure, particularly for repetitive structural arrangements. It also decreases the number of unknowns encountered in the solution of the problem, with a corresponding reduction in computer requirements.

An improved substructuring concept involves the use of the "reduced substructure" technique, in which kinematic constraints are applied on the boundary so as to reduce the number of interaction freedoms. This insures displacement compatibility along the boundaries between adjacent substructures and/or elements.

Some examples of the numerous finite element applications in the marine industry are given in the following sections.

STRUCTURAL ANALYSIS

Discretization is an essential part of the structural analysis problem. In general, discretization introduces approximations into the analysis; for example, a finite element idealization generally involves approximations both of the geometric form of the structure and of the displacements which it develops. The degree of refinement which is employed in the finite element mesh should be based on the analysis requirements, i.e., a fine mesh is needed in regions where the structural behavior is most complex. Moreover, the mesh must be particularly fine if the primary objective of the analysis is stress distribution rather than deflection, because the derivatives of the displacements are represented less accurately than are the displacements themselves when applying the more commonly used displacement method.

The finite element method has been extensively applied to all types of marine structures. The first applications were for the analysis of supertankers (large tankers with deadweight tonnage above 200,000 dwt), as a result of their rapidly increasing size and associated changes in ship configuration. Classification society rules based on previous design and experience were not adequate for these new ships, and considerable structural damage was sustained by some of the earlier supertankers. Computerized structural analyses have provided the means to overcome Rule limitations and to provide the required structural integrity and design efficiency.

Other marine structures such as container ships, ore or bulk carriers, liquefied natural gas (LNG) carriers, submarine structures, surface effect ships, hydrofoils, ice breakers, drilling rigs (fixed and floating), offshore platforms, etc., have been extensively analyzed by computerized methods.

Some examples of these analyses by means of the ABS/DAISY (Displacement Automated Integrated System) program are shown in Figures 1 through 4.

Figure 1 shows the displacements of the deck and the lateral expansion of the front hatches of a container vessel in oblique waves, subject to combined vertical, lateral and torsional loads. Figure 2 shows the extent of a three-dimensional model of an LNG carrier. Figure 3 shows a typical transverse web frame of an oil tanker, its idealization by finite elements, calculated displacements and corresponding stress contours. Figure 4 shows the various design improvements of a shell longitudinal connection to a horizontal oiltight bulkhead girder in a supertanker.
Deck Centerline Vertical and Longitudinal Displacements

<table>
<thead>
<tr>
<th>HATCH NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPANSION</td>
<td>0.382</td>
<td>0.634</td>
<td>0.896</td>
<td>1.25</td>
<td>1.510</td>
<td>2.056</td>
<td>1.989</td>
<td>2.413</td>
<td>1.995</td>
</tr>
</tbody>
</table>

Deck Longitudinal and Lateral Displacements

Figure 1—Overall displacements of deck and front hatches diagonal expansions

Figure 2—Extent of three-dimensional model for independent tank LNG carrier
STRUCTURAL DYNAMICS

Generally speaking, the concepts of structural dynamics are not new; both mode-superposition procedures and direct time-integration analyses of the equations of motion were well understood many decades ago. However, the finite element method provides a unified approach to discretization which can be applied to completely arbitrary and highly complex structures, and the modern digital computer makes possible the routine solution of the resulting equations of motion, which may involve hundreds or thousands of degrees of freedom evaluated at hundreds of time intervals during the dynamic response. Thus, these new tools make it possible to meet current structural dynamic analysis needs in fields such as the design of offshore oil drilling platforms for wave, wind and earthquake loads, the design of supertankers and other ships for wave loads and other hydrodynamic forces, etc.

Some examples of the application of computerized dynamic analysis are shown in Figures 5 and 6.

Figure 5 shows the idealization of an entire cargo ship for the study of vibrations and slam response by means of the SHVRS (Ship-Hull-Vibration Response System) program. Figure 6 shows an isometric view of the model used for the vibration analysis of a supertanker. The model consists of one-half of the entire vessel, and was plotted with the aid of the SAP IV (Structural Analysis Program) plotting routines. Figure 7 shows the wave-induced, springing and combined bending stresses at the midship section of a Great Lakes ore carrier as a function of the wave frequency. This was calculated and plotted with the aid of dynamic analysis programs being developed at the American Bureau of Shipping.
THERMAL ANALYSIS

Thermal stresses and deformations are produced by changes in temperature distribution, and high temperatures influence the mechanical properties of a material to a great extent. Ship structural members and machinery parts are often subjected to various kinds of heat treatment, producing thermal stresses which may result in residual deformation. Temperature differentials of cargo and surroundings, as for example in the high temperature range of hot asphalt (300° F.) or in the very low temperature range of LNG (-260° F.) produce high thermal stresses and deformations which must be considered in the overall design and analysis of a vessel. The magnitude of thermal stresses developed in a hull structure is governed by the restraints provided by surrounding structure and by the non-uniform temperature differences in the hull due to the ship’s internal and external environment.

The finite element method is a powerful tool to obtain thermal stress distributions and deformation patterns. Some examples of its application are shown in Figures 8 and 9. Figure 8 shows thermal loads and calculated stresses on a typical transverse web frame of an asphalt tanker carrying cargo at temperatures of up to
Figure 4—Effect of connection detail on stress concentration

Figure 5—Complete ship idealization
Figure 6—Isometric view of ship model (port)

Figure 7—Dynamic midship bending stresses

Figure 8—Thermal analysis—Asphalt carrier
NAVAL ARCHITECTURE PROGRAMS

Numerous computer programs have been developed over the years to perform various naval architecture calculations such as hull girder shear and bending moment, section modulus, hydrostatic stability and hull characteristics. These calculations all require a description of the hull geometry, usually in the form of offset data. Perhaps the most widely used and comprehensive hull characteristics program is SHCP (Ship Hull Characteristics Program). This program develops both intact and damage stability characteristics for ship forms by conventional methods. Hull girder shear and bending moments can also be calculated. Computer-generated plots can also be produced, such as Figure 10, showing hydrostatic curves, and Figure 11, showing the transverse sections of the hull outline of a vessel.

In addition to comprehensive computer codes such as SHCP, special purpose programs for calculating grain stability, tank ullage, and section modulus are also available to naval architects. As a description of the hull form is a requirement for all of the above calculations, establishment of a common data base for the hull surface geometry is highly desirable.

Computer programs based on the classical naval architecture approach to stability of ship hulls are not easily adaptable to offshore drilling rigs such as the semi-submersible and self-elevating types. These rigs are generally composed of assemblages of tubular sections and/or geometric shapes having planar surfaces. Use of conventional offset data to describe their unusual shapes can be difficult and complicated. To handle drilling rigs, stability computer programs have been developed to perform intact and damage stability calculations for mobile drilling rigs. To overcome difficulties in describing the unusual and multi-connected shapes of these rigs, a typical program such as the ABS-developed DRILRIG contains a library of solid element shapes (cylinders, tetrahedrons, hexahedrons, hemispheres, etc.) which are readily defined with a minimum of data. The user selects the shapes and assembles them so as to represent the compartmentation and geometric configuration of the rig. Stability calculations are then performed by conventional methods.

CONVERSATIONAL PROGRAMS

Conversational programs, which provide for back-and-forth communication between the engineer and the computer, have been successfully applied to obtain ship scantlings (structural properties, such as plate thicknesses and beam sizes) based on classification society rules.

Such a typical computer approach to ship design is the ABS/RULESCANT (RULE SCANTlings) systems of time-sharing programs that determine scantlings satisfying the ABS Rule requirements for midship sections of oil, ore or bulk carriers. The input typed by the user at a time-sharing terminal consists of the basic configuration of the vessel, the plate and stiffener sizes and the location and arrangement of structural members. A data base (file) of all the processed ship information is created for use by the many output subprograms which can be individually selected for execution by the user at the terminal.

One of the programs is used to check scantlings of the midship section. The program determines the Rule-required plate thicknesses and stiffener and hull girder section moduli. It also calculates the weight per unit length, and the allowable shear stress and shear force. The formulas used by the program in determining the Rule requirements are also listed, as a matter of information to the user. After viewing the results, the user can dynamically change any of the input, including the given values of the plating and...
stiffener sizes. Then he can rerun the program while at the terminal and obtain answers within minutes of altering the input.

There is also a preliminary design subprogram, which requires input of only a basic definition of the midship geometry. The program then determines the minimum Rule requirements for the midship section, which can be used for preliminary design purposes.

Many more sections of the Rules will be integrated into a computerized system similar to ABS/RULESCANT.

**COMPUTER GRAPHICS**

With the improvement of the general purpose structural programs to analyze large and complex structures economically, the need for efficient methods of checking input data and reviewing output results becomes more pressing. The field of computer graphics satisfies this need by producing visualizations of the structural
models and stress patterns. The inherent advantages and disadvantages of the two basic methods of computer graphics, passive and interactive, dictate the usefulness and areas of application of each method.

Passive graphic systems include plotters (flatbed, drum, and electrostatic) and microfilm recorders. The nature of these devices precludes user interaction, and therefore they are best suited to applications where a user has time to review the resultant plots before making any changes or going on to the next step. The most popular applications include plots of the geometric model (input) and deflected shapes and stress contours (output). Since turnaround requirements for these applications can usually be measured in hours, the normal operating procedure is to run these jobs in the batch mode on large computer systems and produce tapes which can subsequently run on the plotters.

Most of the large finite element programs have plotting capabilities as part of the basic program or as separate add-on modules. In addition, there are many general purpose plotting programs that can use the output files generated by most finite element programs, although in some cases interface programs must be written. Input geometrical plots (two- and three-dimensional perspective views) and a wide variety of output plots (deflected shapes, force and moment diagrams, stress contours) form the most generally available plot features.

The basic advantage of the passive system is that large amounts of data can be processed economically. The plotter is the only additional equipment needed, and the actual computer runs necessary to generate the plots can be run in the batch mode, with minimal impact on a large computer system. The obvious disadvantage is the lack of interaction. Incremental mode plotters are relatively slow (detailed plots can take many hours), but electrostatic plotters can produce hard copies at rates comparable to most reproducing machines.

Interactive graphics systems consist of display consoles, means of entering and editing data (usually cathode ray tubes, tablets, keyboard devices), and a computer system to maintain the data files and perform the calculations needed to produce the plots.

Interactive systems find their greatest use in design work. The designer is able to communicate with the computer, see the results, and make the necessary changes. The earlier systems required either a totally dedicated medium-sized-computer or a large portion (partition) of the resources of a large computer. In recent years, the availability of time-sharing systems and powerful minicomputers has relaxed these requirements.

The GIFTS (Graphics-oriented Interactive Finite Element Time-Sharing Package) system, designed primarily for ship structures, is one widely used interactive package. The entire system accesses a Unified Data Base (UDB) which stores all pertinent data on a set of random access files. Each individual module can access and operate on the UDB. After the entire model has been verified, part of the UDB forms the input to a general purpose analysis program such as DAISY, NASTRAN (NASA Structural Analysis), or SAP IV. The output from the analysis program is then incorporated into the UDB and additional modules can display results.

Some of the displays obtained during the various phases of the analysis of a vessel are shown in Figure 12.

**INFORMATION RETRIEVAL SYSTEMS**

The service experience of vessels at sea provides a wealth of data to evaluate ships. Information on occurrences of structural and mechanical damage and failures must be collected and statistically analyzed. Subsequent feedback of this information will enable shipbuilders and designers to incorporate this knowledge into future improvements in design, construction and analysis techniques. The only means to efficiently handle the vast amounts of data involved is through information handling systems.

A comprehensive computerized system for information storage, correlation and retrieval has been implemented at ABS. This system, known as ABSIRS (American Bureau of Shipping Information Retrieval System), handles data concerning shipowners, shipbuilders, ship characteristics, service histories and
other pertinent data relevant to merchant vessels of the world. ABS uses its vast stores of information and worldwide telecommunication facilities to provide easy and rapid accessibility to the industry.

The ABSIRS system consists of seven files, whose functions can be summarized as follows:

Master File is the nucleus of ABSIRS. It contains all the data in the published RECORD of the American Bureau of Shipping, a ship registry that contains pertinent characteristics of more than 42,000 vessels—virtually all sizeable vessels in existence—including the more than 13,000 vessels that have been classed by ABS.

Technical Notes File contains service data limited to statistical compilations and correlations of damages or casualties in vessels classed with ABS since 1965. Hull or machinery failures that are considered significant are entered in the file, i.e., those failures that may eventually show a recurring problem and thereby prompt Rules changes or revisions in existing construction techniques. The File contains a brief description of hull damages according to type (buckling, welding, cracks, corrosion, etc.).

Construction File stores additional information on vessels of ABS classification, including particulars on characteristics of hull construction and materials and machinery items and associated components.

Dead File stores data on ships that have ended service life. The recorded data passes from the Master File to the Dead File, insuring a preservation of the vessel's history.

On-Order File carries data on vessels in excess of 1,000 gross tons that are either being constructed or are under contract to be built in shipyards around the world.

Owners File is a record of the names and addresses of owners, agents, and operators of ABS-classed vessels appearing in the RECORD of the American Bureau of Shipping.

Shipbuilding and Drydock File lists the names, locations, capacities, and descriptions of shipbuilding, drydock, and repair facilities available throughout the world.

Each file can be searched separately and there is a multi-file capability that allows the information within each file to be cross-correlated with data in any other file or files. Inquiries may be based upon any category stored in the computer, and many report formats are available to suit the user's needs.

CONCLUSIONS

Computer usage in the marine industry is extensive, diversified and expanding. It combines the use of mathematical techniques with the latest technical facilities of computers, software, hardware and communication with the user.

The developments of computer analysis in the industry point to the following future trends:

1. More intelligent use of the computer as an engineering tool will result in more economical analysis methods. For example, advanced techniques such as substructuring will find increased applications in the design and analysis of ship structures.
2. Increased use of computer graphics display terminals will facilitate the automatic generation and verification of the large amounts of data necessary to create an extensive finite element model.
3. Engineering analysis programs will acquire a more interdisciplinary character. For example, the hydrodynamic forces acting on a ship will be calculated by the same computer program that determines the resulting structural response.
4. More interactive programs emphasizing user-computer interaction will result in a greater variety of applications among a wider spectrum of users.

REFERENCES


Evolution of automation in terminal air traffic control

by HOWARD R. McGLAUFLIN
Federal Aviation Administration
Burlington, Massachusetts

ABSTRACT

The Evolution of Automation In Terminal Air Traffic Control was written to provide the reader with an overview of the why, when and how of the needs of air traffic controllers in the terminal. This paper compares systems and a decision to award contracts. This paper contains a description of the subsystems used at ARTS III facilities and its capability for expansion. Attachments to this paper show a radar display, an ARTS III display, a system design and a work flow-through design, basically how it works.

The intention of this paper is to provide an overview of the evolutionary process of system automation in the Terminal Air Traffic Control environment, and provide a look at the basic flow of data through the existing system.

To do his job in the 1920's, the controller had only to wave a green or red flag; in the 1930's he operated a radio in a control tower; in the 1940's he had to learn to operate a radar set; and now he must start learning to communicate through a computer.

The need for some degree of automation to assist controllers in Terminal Air Traffic Control has been apparent for some time. Air Traffic volume has been increasing and is forecasted to increase so that many terminals will be taxed to capacity. Delays at the busiest airports are frequent, and all too often, lengthy.

The formal initiation of plans to develop an automated system for terminal traffic control occurred in March 1961, when the Project Task Force was established. The result was the Project Beacon Report, which was submitted to the FAA (Federal Aviation Administration) and then to President Kennedy in September of 1961. One significant recommendation of this report was, "utilization of general purpose digital computers to provide controllers with aircraft position information."

Subsequently, the FAA's System Design Team was established to fulfill the requirements of the Project Beacon Report; a functional specification for an experimental model of an automated radar terminal system that could be used for appraisal of the concepts in a field environment was developed. In 1963, the FAA awarded a contract to Univac to provide the computers, software, and system integration efforts for the establishment of this model known as Advanced Radar Tracking System I, in the Atlanta, Georgia, terminal. After field testing, this system was implemented and has been in use since 1966. Basically, Advanced Radar Tracking System I, utilizes a general purpose digital computer together with video digitizers and displays to increase the radar and beacon video on a controller's console. It does so by the display of alphanumeric aircraft identity and altitude information which is automatically associated with the proper video returns.

The next activity involving terminal air traffic control was the implementation of an automation system in the New York Common Instrument Flight Rule Room. This room was implemented in 1968, located at Kennedy Airport in New York City, and permits the individual airspaces over Kennedy, LaGuardia, and Newark Airports to be combined into a single operation by utilization of computer assistance. This system is called Advanced Radar Tracking System IA, "this system differs from the Advanced Radar Tracking System I in that it uses two separate, non-collocated radar and beacon inputs."

In 1969, the FAA arranged for implementation and demonstration of a terminal automation system suitable for medium traffic load in Knoxville, Tennessee. The purpose was to demonstrate that a small, stored program computer offers more flexibility and capability than special purpose calculating devices, which had been devised for the detection and display functions at smaller installations. After these hardware implementations were carried out, in December of 1969, a committee was organized by the Department of Transportation (commonly called the 1980 Committee) Air Traffic Control Advisory Committee whose duty it was to project air traffic requirements until the 1980 and 1990 time frame. This Committee report was released and is now being used as guidelines for automation development and enhancement. One major recommendation of the 1980 committee's report affecting terminal automation was, "The addition of more capability
to the terminal automation system in order to provide such functions as command and control sequencing, conflict detection, and collision avoidance, and other functions which can increase safety or maximize terminal system aircraft acceptance rate.3

Based on experience gained from the operation of ARTS I and IA, as well as continuing analysis of the present and projected air traffic situation, the FAA developed a design for an improved air traffic control system. The Advanced Radar Tracking System III, ARTS III, was designed to simplify the acquisition and maintenance of radar identification, display beacon derived altitude data, simplify intrafacility and interfacility coordination procedures and reduce the communications workload. (See Attachment #1) The result is a more efficient utilization of terminal airspace and air traffic control personnel and an enhanced safety system. The ARTS III system is now being implemented at 62 major air terminals, and at the FAA Academy in Oklahoma. It is intended to be a first step for many in terminal automation of air traffic control and to provide an automation basis on which to build. In order to provide the capability for growth of system functions at the same pace with increased requirements and resultant developments, ARTS III was additionally designed to be built with a modular expandable concept. Because software is, of its own nature, modular expandable when using the executive program/sub-program hierarchy, and also dependent on the hardware modularity, the primary design considerations were related to the hardware. Being an add-on system, basically, the major sub-systems were added to the existing terminal area equipment: Data Acquisition Subsystem (DAS), Data Processing System (DPS), Data Entry and Display Subsystem (DEDS).

The following is a description of the three sub-systems used in the ARTS III system:

(1) The ARTS III DAS, which will accept inputs from a variety of airport surveillance radars and beacon interrogators which consist of the azimuth, range and timing group, the beacon reply group, and the azimuth pulse generator. The azimuth, range and timing group generates all the basic timing pulses for use by the DAS. The beacon reply group detects and interprets the beacon video signals; the azimuth pulse generator, physically mounted to the radar pedestal, converts radar antenna position to digital position data.

(2) The DPS consists of the data processor (IOP) and its peripheral equipment, as well as the operational computer program. (See Attachment #3) The IOP receives digitalized beacon target report messages from the DAS. It also receives flight plan data from a computer in the enroute center via the ICA. The IOP also receives controller-initiated messages from the DEDS. The IOP provides the system with a capability for arithmetic computation, logical decision making, data processing, and overall system coordination. Operation of the IOP is controlled by a stored program located in the memory bank with each major processing task organized into a sub-program. The IOP then employs an executive control sub-program that without performing any processing tasks itself, serves to control the execution as needed, rather than in fixed sequence. The task selected depends upon an assigned priority scheme that adapts to changes in the system processing load and permits the processor to respond to simultaneous external demands. Some tasks are executed on the basis of a fixed time interval, while others depend upon the completion of prerequisite processing of external events. The IOP provides control of overall communications between the DAS, the DEDS, and the peripheral equipment via high speed digital input/output channels.

The basic DPS includes peripheral equipment commonly found in a data processing application:

(1) A magnetic tape unit provides permanent storage for the computer programs, and is used to load the programs into computer memory. In addition, selected data obtained during system operation may be recorded on magnetic tape for future processing and analysis.

(2) An input/output console containing a low speed printer, typewriter and keyboard, and paper tape facilities is provided for controller communications with the IOP. This console is used off-line to enter variable parameters required by the system and is used as a back-up device to load programs when the magnetic tape unit is unavailable. It also provides on-line printout capabilities for various sub-programs. The hard copy may provide alarm, recording on requested functions. Alarm printouts result from program-detected malfunctions in hardware and errors in input.

(3) The DEDS consists of a common equipment assembly, display consoles, and data entry sets. The DEDS provide the man-machine interface between the air traffic controllers and the automation equipment.

Since the ARTS III system has been in operation at Chicago O'Hare Field since May of 1971, many questions have arisen and one point has been made very clear, "the automation applications made to date in the air traffic control system do not seem to have increased the capacity and productivity of the system, even though there are some indications of increased safety and relief in the workload of the controller."4

This paper has attempted to provide an overview of the evolutionary process of system automation in the Terminal Air Traffic Control environment.

BIBLIOGRAPHY

Tracked targets, indicated with alphanumerics, may be offset to any 45° direction. Untracked targets are always offset to NE.

Attachment #1—A Representation of the ARTS III Video Screen (Alphanumeric Display)
1. NLM 1411: Tracked handoff from Controller A to Controller F
   (Format blinks after F accepts)
2. TWA 70: Tracked by Controller A full data block displayed.
   (Alt: 1400 ft., ground speed 250 kts.)
3. QAN322: Tracked by Controller A, Radio failure (RF blinks)
4. BAO148: Tracked by Controller E at enroute center, handoff from
   Controller E to Controller A (format blinks).
5. JAL 103: Tracked by Controller B, "Quick Look" by Controller A
   caused the data format to appear.
6. Coast/suspend list - provides identification of aircraft under
   control of Controller A but which are not currently being tracked.
7. "CM" : tracked by Controller C no data format appears since
   target is being controlled at another display.
8. "M6" : tracked by Controller A target has been assigned temporarily
   to a "suspend" status as indicated by line identifier "M".
9. System Data Area - Provides current time, altimeter setting, selected
   beacon codes untracked emergency/radio failure indicators and
   memory readouts.
10. Preview Area - Provides keyboard entry characters and controller
    requested data (e.g. flight plan data).
11. * Untracked, limited data block.
12. Arrival/Departure List - Provides identification of aircraft which
    are scheduled to arrive at the entry fix or depart the airport
    within a few minutes and will be controlled by Controller A.
13. Untracked, emergency (format blinks).

Attachment #1A—Legend of Alphanumeric Display

Non-Alpha-Numeric Display

LEGEND:
- Primary Radar Return
- Beacon Radar Return
- Emergency Beacon

Attachment #2—Non-alphanumeric Display

Attachment #3—Block Diagram of DPS and related equipment
prepared by H. R. McGlauffin, Plans and Programs Specialist,
Air Traffic Division, Burlington, MA
The following illustration shows the basic flow of data through the system.

Attachment #4—Basic Flow of Data Through the System, prepared by T. Pastore, Secretary, Boston Tower.
Computer graphics in an automatic aircraft landing system*

by E. H. REITAN
ITT Gilfillan
Van Nuys, California

and

S. H. SAIB
University of California
Santa Barbara, California

ABSTRACT

The Marine Air Traffic Control and Landing System (MATCALS), being implemented by the Naval Electronic Systems Command, provides advanced capabilities for fully automatic all-weather landing. Using radar-derived aircraft position reports, a ground computer provides appropriate guidance commands, which are transmitted to the aircraft's autopilot, and used to fly the aircraft automatically to touchdown.

Due to severe system requirements of one-half minute landing intervals, and six aircraft simultaneously on final approach, a unique combination of display presentation and operator interaction techniques must be used. These support ground operators who as controllers are responsible for initialization of each landing sequence, monitoring its progress, and aborting the sequence if an unsafe condition develops.

A display concept has been developed for MATCALS with the goal of reducing the controller's workload and increasing his effectiveness. The display is used as a single working surface for both output and input functions. A dynamic, graphical display format presentation, with alphanumeric annotation, and alert information is displayed for the operator in multiple colors. More than 150 system controls are organized on the same display as a highly structured hierarchy of virtual control buttons grouped into menus. The operator is prompted through all data entry and control sequences. All operator entry is made by using a "Rand" type data tablet.

MATCALS OVERVIEW

A simplified block diagram of MATCALS is shown in Figure 1. MATCALS is organized into three main system segments: the Air Traffic Control Segment, the All-Weather Landing Segment, and the Control and Central Segment.

The system normally operates from the Control and Central Segment which receives information from the two other segments. The Air Traffic Control Segment provides automated surveillance and traffic control within 60 miles of the airfield. Specifically, MATCALS provides for surveillance, identification, tracking, sequencing, vectoring, and inter-facility coordination for all approach, departure and overflight operations within the terminal area.

The MATCALS Landing Segment provides for fully manual to fully automatic landings. The sensor for

---

*The comments made herein are those solely of the authors since they pertain to early research and development and do not necessarily represent exact Government or technical requirements of the United States Marine Corps and the Naval Electronic Systems Command.
the landing segment is the AN/TPN-22 radar system shown in Figure 2. This precision approach radar provides the necessary positional reports with sufficient accuracy for automatic landings.

Three modes of landing are available to an aircraft. In Mode 1, ground derived steering commands are transmitted over a data link and directly introduced into the aircraft autopilot.

Under Mode I, the system is able to transmit course deviations to any cross pointer equipped aircraft via a variety of data links. Specific up-links which have been implemented include pseudo-ILS signals receivable on standard airborne ILS equipment and side-band coded signals on a UHF voice channel. Pseudo-MLS signals receivable on the planned standard MLS airborne subsystems will be implemented later.

In Mode III, ground derived approach course deviations are displayed to the landing controller who verbally transmits the landing correction commands to the pilot. The Landing Segment also provides for maintaining inter-aircraft separation minimums, monitors the approaching aircraft for acceptable positioning within defined boundaries, and provides for automatically aborting unsafe landing sequences.

CONTROLLER DISPLAY DESIGN OBJECTIVES

Emphasis has been placed on the selection of the computer graphics display formats and the type of operator interaction mechanisms which will support the MATCALS Landing System display requirements.

The display system must provide for the monitoring and control of up to six simultaneously landing aircraft. Inherent in the concept is the facility for controllers to assume responsibility for additional aircraft control in the event of failure of other display consoles. The overall objective is reduction of controller's workload and increase of landing safety in a multiple landing aircraft environment. A number of assumptions were made and desirable features were identified concerning characteristics of operator abilities, the varying difficulty of the operator's task during downgraded conditions due to hardware failures and reduced manning support, and the operational mission of the MATCALS production system.

It is first assumed that the console users in general will have little or no computer training. It is therefore unrealistic to impose any unnatural input techniques or display format output techniques. The operator/intermediate interface must be as natural as possible and tailored to his conception of the landing operation—not just to simplify computer requirements. Ideally, representation of data or conditions should be done pictorially using a situation display type of abstraction. Auxiliary information may be presented alphanumerically to reinforce the pictorial representation. This auxiliary data assists the operator with his evaluation of the present operational situation. It is felt that graphical presentations lend themselves to rapid qualitative operator evaluation of global relationships of the data, while alphanumeric data presentations amplify and quantify the details of the landing situation. Both types of presentation (viewed as hierarchy of information) are necessary, and are most desirable if presented simultaneously on the same display surface.

The parallel presentation of pictorial (graphical) and tabular (alphanumeric) information simplifies not only the learning of the system but reduces the reaction time during critical and alert periods of system operation. The operator should not be required to divert his eyes to other presentation panels in order to make decisions, instead he should have available on a single display surface all pertinent information within his field of view. This allows him to assess the situation and make a corrective decision.

The complexity of the display format must be kept at a minimum. The state of the landing mission environment is presented as directly as possible. Information should be non-redundantly presented to the operator to assist and not hinder his evaluation of the situation.

All input mechanisms should be as direct as possible. Operator input should not be alphanumerically in nature (which is harder to understand being another level of abstraction). The MATCALS operational environment
cannot tolerate lengthy input sequences such as from
a physical A/N or function keyboard. Inputs instead
should be implemented by operator selection of com-
mands and controls from a list of alternatives pre-
sent on the display. This list must contain the com-
plete set of controls for system operation. The range
of his possible input responses to the output data may
be limited by the computer system to those which are
correct for that instance of time. The computer there¬
fore guides the operator through entry sequences and
prompts him as necessary for correct sequences of data
entry. Menu groups of virtual control buttons are
organized as a hierarchy into a tree. The operator
progresses through the branches of the tree for all
control and data entry functions. At any point in time,
the interaction sequence is clear to the operator. In
addition, the interaction method of implementation
should be flexible to system enhancements.

Both output and input functions are performed
through a single display working surface. All input
controls are immediately adjacent to the output infor-
mation from which a control decision must be made.
Also, input controls are immediately available for
operator actions. This varies from the conventional
approach in that it eliminates additional hardware con-

rol panels and switches which add to the system com-
plexity. The operator does not have to divert his atten-
tion from the single working surface display to search
and locate other hardware controls.

Finally, the display must be responsive to operator
requests. This requirement is satisfied by a careful
design of both the hardware and software system. The
level of responsiveness may vary, however, with the
operator's evaluation of the complexity and importance
of the requested service. As an example, the execution
of a command to abort a landing sequence (a wave-off)
should be immediately serviced and feedback acknowl-
edging the action returned to the operator in a frac-
tion of a second. Conversely, the request for a change
of the display scale range may be delayed for several
seconds with little operator dissatisfaction.

SYSTEM CONFIGURATION

Figure 3 is a simplified block diagram of the MAT-
CALS System at Patuxent River, Maryland showing
major hardware devices. The system is configured
around a Univac AN/UYK-7 general purpose computer
with 48K words of memory (32 bit words, 1.5 μs cycle
time). The AN/UYK-7 is a medium scale, single ac-
cumulator machine which has floating point hardware
and four independent data channels. Software for the
AN/UYK-7 computer was developed in the CMS-2
language operating under an executive configured from
the Common Program.

The central computer receives aircraft position from
the precision approach radar, an ITT Gilfillan AN/
TPN-22 radar with a nominal coverage of 10 miles in

range, 46° in azimuth, and 8° in elevation. The radar
system itself contains a minicomputer and provides
stand-alone capability for Modes II and III landings
in the event of a failure of the central system.

The AN/UYK-7 computer calculates the flight path
corrections and transmits the results to the aircraft
via the digital data link. These corrections take the
form of pitch and bank commands which are fed
directly to the aircraft's control surfaces.

The IMLAC display computer with its associated
CPS-8001 color slave display console and Computek
data tablet constitutes the primary interface between
the operator and the landing system. The IMLAC
computer is a single accumulator machine with 16 bit
words, 16K memory, 1.8 μs cycle time. The data tablet
is utilized as the main interaction device. Unlike the
conventional trackball, joystick, and lightpen, constant
monitoring of the device by the computer cursor track-
ing calculation is unnecessary. There is no loss of
track problem as with light pens, no wrap around
problem as with trackballs, and no slow positioning as
with trackballs and joysticks. Both functions of posi-
tioning and selection are provided by the stylus and
tablet combination.

The division of responsibility between the central
landing computer and the remote display computer
was designed to

1) reduce loading on a saturated AN/UYK-7 com-
puter

From the collection of the Computer History Museum (www.computerhistory.org)
(2) minimize the message transfer rate between the AN/UYK-7 and IMLAC computers
(3) provide for functionally partitioning display/interaction tasks from the landing guidance computer to allow easy extension for future multiple display support.

DISPLAY DESIGN

Design features

Based on the design objectives of the previous section, techniques were introduced which reduced the complexity of the display, assisted the operator in his interaction with the system, eliminated keyboard entries, and increased operator acceptance of the display format. These features were added without any degradation to the basic system requirements. The operational requirements for the MATCALS mission state that up to six aircraft may be simultaneously on final approach. The design therefore, has been predicated on a combination of display output and input techniques to allow the monitoring of the maximum number of aircraft by one operator, and a desired attainment of the control of six Mode I landing aircraft by a single operator.

Although it is doubtful that the single operator situation is desirable (with probably one operator for each two aircraft on a final approach being optimum), it is still mandatory that an operator have cognizance of all other aircraft on approach for safety reasons. The relationships of the aircraft under his control with those of other controllers must be continually monitored. In addition, it may also be postulated that due to hardware failures of other display consoles (and computer sections), or insufficient console manning resources, it is necessary that each operator can assume responsibility for more aircraft than is usual. This type of inherited fail-safe protection is a spin-off of the six aircraft status display requirement. The design of the MATCALS landing display format was therefore made with the assumption that data for all six aircraft on final approach be simultaneously presented, and that any individual operator can assume control of any or all aircraft.

Obviously this specification places an inordinate burden on the operator if conventional display and hardware techniques are utilized. Figure 4a depicts a conventional display which is used in the landing of a single aircraft. One could imagine the complexity of the hardware control panel if it were expanded to allow the landing of six aircraft. A system of this complexity would deluge the operator with controls, displays, and possible operational input combinations and confusing input sequences. On the other hand, Figure 4b shows the display hardware used in the MATCALS system to land six aircraft with just the single display surface and the tablet.

Display format design philosophy

The simpler hardware is possible due to the implementation of techniques wherein the computer organizes the type and format of data graphically or tabularly presented to the operator. The operator executes all data entry and request sequences to display auxiliary information, monitors the operation of the radar and computer hardware and software, analyzes and observes landing sequences, and is notified of any abnormal alert or danger conditions.

Four distinct types of information are presented on the display CRT face (Figure 5):

- Graphical—Presentations with amplifying alphanumeric information
- Tabular Parameter Data
- Alerts and Status Information
- "Virtual" Pushbutton Menus

Up to now, situation data of radar video presentations have been the conventional and common method of information presentation in aircraft control and aircraft landing display systems. To clarify the data on the screen, graphical and pictorial information has been added in some cases, often annotated by limited amounts of alphanumeric information (such as tracking data blocks for air traffic control systems).

The display format implemented for the MATCALS effort extends these concepts. In addition to graphical information, critical landing system parameters are presented grouped as tabular listings of alphanumeric data. Alert messages of hardware or software malfunctions and/or landing system safety irregularities are presented as necessary along with system status information as requested by the operator. Instead of hundreds of hardware control buttons continually being allocated space on a panel, the computer dynamically presents "virtual" control button panels which are displayed on the face of the CRT only when they are necessary and removes the panels when their use is no longer required. For example, it is necessary to have the controls which initialize various parameters to define an impending landing sequence (such as entering aircraft type and aircraft tail number) only at specific time intervals during the landing sequence. At other times these controls are extraneous. The display to be implemented as part of MATCALS utilizes this technique of control panel variability to simplify input operations and to reduce display format complexity.

The MATCALS Display presents all system status and controls in a single basic display format to the operator. Extending this concept to overlay the display graphical situation data with background air-search radar or precision approach radar video is practical using time compression display techniques. Whereas older conventional displays commonly used round CRT's, the MATCALS display organization, using multiple output data types, calls for a rectangular display.
Figure 4—Evolution of landing controller operational consoles
a—Older conventional console for a single aircraft system

Figure 4b—Interactive computer graphics console for a six aircraft system
format (21 inches is sufficient). The use of a round CRT compromises the optimum positioning of the tabular parameter, alert messages, status data, and virtual control buttons. Air traffic control displays which use round CRT’s often place short tabular preview lists in portions of the situation surveillance area—and thereby obscure and mask possibly important data.

CODING MODALITIES

A number of coding approaches are being utilized in the MATCALS display format and interaction sequence. The modalities of color, intensity, line type (dot-dash), shape, blinking, and sound have been used to increase the range of coding dimensions, simplify the display format, and attract attention to specific information on the screen (Figure 6).

Burdick\(^2\) has published the number of identifiable coding levels associated with these modalities:

<table>
<thead>
<tr>
<th>Modality</th>
<th>Coding Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>3-10</td>
</tr>
<tr>
<td>Intensity</td>
<td>2-4</td>
</tr>
<tr>
<td>Shape (abstract)</td>
<td>8-16</td>
</tr>
<tr>
<td>Flicker (Blinking)</td>
<td>2-4</td>
</tr>
<tr>
<td>Line Type (dot, dash)</td>
<td>3-4</td>
</tr>
</tbody>
</table>

These modalities allow additional data to be displayed and absorbed by each operator so as to minimize the number of consoles and operators.

Color

The most intriguing modality, color is provided by the use of a slave beam penetration tube display, connected to the IMLAC. Assignment of display entities to the displayed four colors is as follows:

GREEN  All background information, parameter tables, time, wind, axes and labels, status, and control pushbutton menus.

YELLOW Aircraft and higher priority entities,
diamond, and inverted triangle). These symbols are displayed on the Az-El surveillance area portion to indicate relative position, and displayed on the wave-off button and parameter list entry rows to allow rapid correlation and association of information by the operator.

**Flicker**

A combination of color, sound, and flicker was used to present alert messages. Blinking of messages annoys the operator. Red, while attention gathering, forces the operator to divert his eyes even after the message has been sensed. Inspired by Kubrick, the display of a new alert message is initially red with a rapid flicker accompanied by a low pitch buzz sound. It then automatically changes to a silent, steady state red. When the operator first acknowledges the message, the color changes to a less annoying green. Subsequent acknowledgment deletes the message from view. Acknowledgment is accomplished by merely pointing to the message with the tablet pen.

**VIRTUAL BUTTON CONCEPT**

The graphical equivalent of hardware control buttons is presented on the face of the display. The operator may point to these “virtual” control buttons and the corresponding control function will be executed through the computer software.

For the MATCALS Display Console, a “Rand” X-Y data input tablet is used as the primary input device. The operator holds a stylus over a tablet surface horizontally mounted below and in front of the display CRT. This stylus can be randomly pointed to any position on the tablet surface. The computer system may read the X-Y position of the stylus over the pad (even when the operator holds the stylus some distance above the surface) and generates a cursor symbol (a small cross) on the face of the display. As the stylus is moved, the cursor on the display tracks the operator’s hand-held stylus position on the pad as a feedback mechanism. A micro switch in the tip of the stylus is activated when the pen is pressed to the surface of the tablet. Therefore a button “push” is executed by the operator moving the pen to position his cursor over the “virtual” button he desires to push, and depressing the stylus. The operation is analogous to “pushing” a hardware button. This input strategy is very natural and the operator becomes immediately accustomed to its use.

**HIERARCHICAL ORGANIZATION OF COMMANDS**

The MATCALS Display organizes the operator’s data input and system controlling actions into cohesive and logical multileveled groups. Each group consists of a number of menus of displayed virtual control buttons.

The hierarchical ordering of button menus allows the manipulation of commands and information requests. The command groups are ordered into a tree structure of usage and inter-relations. The user traverses branches of the tree (commands) as he progresses through the structure. This model attempts to describe the control relationships identical to the user’s conceptual understanding and to guide him through his interaction with the control structure. Normally only classes of important control are displayed. When the user selects a class of commands, the detailed control types of sub-menu commands appears. When the user completes his interaction and use of these commands, he may return back to the higher levels of control types and command classes. The lower level is then automatically removed from the screen. We therefore have dynamic display and removal of command pushbuttons depending upon their syntactic requirement during an entry sequence. Extraneous and unnecessary controls are not continually displayed. A particular button menu is displayed only when its requirement is logically necessary. Any tendency toward a complex, cluttered, and confusing control panel presentation is avoided. The user can always quickly return to a higher command level even before making an actual entry at that level by pushing a “RETURN” button (which is always present). He may also directly return to the top level of the command class menu of virtual buttons when desired. All controller interactions with the system are checked for errors and provide feedback as to the entered quantity or function.

**Interaction description**

Figure 7 shows a primitive controlling initialization sequence to define an impending landing sequence. When he pushes the button “Select Landing Sequence,” a new set of buttons appears. The operator may manually define the glideslope (and be taken down another branch where he enters a numerical value), enter the radar acquisition position, or (as is shown) select the desired Landing Mode. When this is pushed, the menu list allows him to select modes I, II, or III. As there is no computer communication with the landing aircraft in mode III, there is no need to enter a data link address and he is returned to level B and allowed to continue his data entry. If I or II is selected, he is allowed (at D) to enter an octal data link address. If he wishes to do so, a graphical numerical keyboard appears. In sequence, he may point to the desired keyboard buttons and the entered numerical value is accumulated. When he is satisfied with the value, he enters it by pushing the return button.

If no entries are made at a particular level, previously defined default values are assumed and used by the system.
This approach to computer input takes full advantage of the graphic display and of the human eye's ability to scan rapidly a single menu of commands. The entry rate is higher than by using a multi-function keyboard or array of hardware pushbuttons. The slowness associated with these hardware devices is due to the finite search time required to find the desired button and diversion time of the operator's eyes away from the primary display surface. The user does not have to recall a specific input control response (as with a keyboard) as all allowable responses are displayed. All displayed system status information necessary to make a decision is adjacent to the selection area.

Two different approaches to the ordering of levels must be mentioned. Generally, for initialization of a number of parameters a sequential multileveled ordering is optimum (a tall tree). Each group has a few selection buttons in each level menu. This forces the user to step through all the control initialization options to enter either the required data or to acknowledge that the system default value may be assumed (when he elects to go on in the hierarchy sequence without a data entry). This insures that an operator decision was made on all critical system parameters.

However, for modification of previously entered data, the scheme above would be frustrating to the user if he had to go through all parameters to get to the one he wished to modify. So instead, a shallower short tree of fewer levels is used for this case. However, each level has many buttons in each menu to allow the selection of the entity to be modified in a parallel fashion from a large list.

MATCALS utilizes two distinct multi-level trees:

a. Wave-Off Initiation Buttons
b. System Control and Parameter Input Buttons

The entire tree data structure is shown in Figure 8.

The System Control and Parameter Input Tree is the most complex example of hierarchical interaction with up to 6 levels of menus. Typically, the operator is lower than the third level only for brief times. The controls are grouped into six distinct classes:

- a. Landing Channel Definition (aircraft initialization)
- b. Landing Procedure Initialization
- c. Landing Mode Selection
- d. Radar Track Acquisition
- e. Systems Controls (data recording and simulation)
- f. Display Controls (range scale and other display format selection)

Within these classes are the levels of sets of virtual button menus that control the system or are used to enter system data. There are 2-6 buttons in each menu set. At many stages of data entry, general purpose menus are presented in the general scratch pad area to perform:

a. The entry of numerical values (using a virtual graphical keyboard)
b. The selection of an aircraft track number
c. The selection of a general purpose index

d. These general purpose menus may be thought of as interaction subroutines which may be used in any place in the control/data entry hierarchy. These graphical interaction subroutines appear only when required.

The use of hierarchically organized dynamic command panel menus in the MATCALS display system provides:

a. insured accessibility to information—fast update, reduced delay time, and improved spatial accessibility (no longer the need to search for that one correct button in a haphazard fashion).
b. increased reliability from the use of software to implement these complex control relations instead of the multi-wire/hardware switch approach.
c. increased flexibility—easier to adapt to required changes in types of controls.
d. increased utilization and efficiency of the control actions—improved vigilance of the observer—increased data input rate for untrained operators.
e. Commands may be tailored to the system understanding and educational level of the user/operat-
tor. The commands are organized to allow the operator to teach himself the man/machine interaction.

**Human factors considerations**

Some quantitative research has been made on hierarchical dynamic controls using displays. Uber has published that a maximal response time to this type of request is 0.5 seconds, which is one of the reasons for the high level of intelligence utilized in the MATCALS local display processor. Uber has also analyzed the optimal number of menu selection buttons per level. With a typical scan time of 0.5 seconds per entry, and selection time of 2 seconds per entry, the menu length is 7.7 entries. Experimental and trained users can have practical menu lengths of 10 to 40. Because of the possible background of the MATCALS user, a maximum menu length of 6 buttons has been selected.

**DISPLAY FORMAT**

**Introduction**

The display is divided into the following general areas: (Figure 5)

- Az-El Display
- Aircraft Parameter Table
- Wave-off Buttons
- Alert Message Area
- Interactive Control Buttons
- Width-Height Display

The Az-El area is the primary observation area. Glideslope angle and centerline, acquisition gate positioning, aircraft positions (6), and associated tracking data blocks are graphically represented. Three scales are selectable. Wind and time of day information is also presented. Aircraft positions on the display are rapidly updated to show relative motion of the aircraft.

The aircraft parameter area is a tabular alpha-
numeric listing of detailed information associated with each aircraft. Its use is considered secondary to the Az-El area, and more of a reference to less often needed data. The necessity for each type of data to be displayed for each aircraft will be evaluated during actual landings.

Wave-off buttons are continually displayed so they may be immediately accessible to the operator. Alert messages are presented in the lower corner of the Az-El area where they are immediately noticed.

The menu of hierarchical interactive virtual pushbuttons is presented on a row adjacent to the Az-El area for easy reference.

A Width-Height display (WHI) is presented at the right of the lower portion of the display. It is of fixed scale, and only one aircraft is shown. The WHI is primarily used for only the terminal portions of a landing sequence near touchdown.

**Detailed description**

A detailed description of the major display format areas as shown in Figure 9 is given in the following paragraphs.

**Aircraft parameter table**

Each aircraft is allocated one line in the aircraft parameter table located across the top of the screen. Each line of data is identified by the aircraft index and the associated symbol to be plotted on the Az-El and WHI displays. The aircraft parameter table contains alphanumeric data associated with each aircraft, including static parameters describing the particular aircraft and dynamic values related to the aircraft’s position and movement. The static quantities remain fixed throughout a track while the dynamically varying data is updated every half second. Static quantities are side number of the aircraft, aircraft type (A7, F4, etc.), communication channel address, landing mode (I, II or III), and glideslope angle.

Dynamic quantities displayed alphanumerically are pitch and bank angle commands being transmitted to the aircraft (Mode I), sink rate (fast/sec), speed (knots), range (tenths of miles), height (feet), height error from the desired glideslope (feet), azimuth position (feet) and time-to-touchdown (in seconds). Corresponding parameters for each aircraft are arranged in columns and are appropriately labeled.

**Wave off area**

A row of wave off initiation buttons is placed immediately below the parameter table. This position puts the wave off area in close proximity to both the parameter table and the Az-El portion of the display. The wave off area is maintained separate from other interaction functions to allow continual and immediate availability to the operator. To speed up the wave off process, each aircraft is provided with a separate button, identified by the correlated aircraft index and symbol. When a wave off button is selected the operator is then required either to confirm that a wave off is to be generated or cancel the wave off request. Confirmation of a wave off request is required to lessen the probability of accidental pushing of a wave off button.

**Az-El display**

The Az-El display is centrally located and occupies the major portion of the display area. Aircraft height versus range is presented in the upper half of the Az-El display area and lateral position versus range in the lower half.

Provision has been made for a selection of three scales. Twelve mile, six mile and 1.2 mile ranges are selectable with a corresponding change in the vertical and lateral scales. Logarithmic or expanded linear scales are desirable alternatives that will be considered for future implementation. Axes are labeled for ease in estimation of aircraft positional relationships and the touchdown position is offset from the edge of the display to allow tracking for a short distance after touchdown.

A unique identifying symbol is used to plot each aircraft’s position. The symbol is also shown in the parameter table to correlate the tabular and graphical presentations. In addition, a tracking alphanumeric data block is displayed adjacent to each aircraft position, connected to the aircraft symbol by a short leader. The aircraft index, side number, landing mode, simulated target tag, range, velocity and vertical and lateral glideslope errors can be read directly from Az-El display without consulting the parameter tables.

The time-of-day, wind speed and heading, displayed glideslope angle, and runway heading are listed in unused portions of the Az-El display area. The time is updated every second and is an indication to the operator that the control computer is active. High priority alert and warning messages appear in the lower left corner of the azimuth display.

**Interaction and monitor area**

The lower portion of the screen, approximately one fourth of the total area, contains the interaction functions. All operator/system interactions except wave off initiation and positioning the acquisition gate directly on the Az-El display are accomplished in this part of the display.

The function selection menus consist of several sensitive buttons that can be selected by positioning the tablet pen within the button boundaries. The display processor reads the pen position and determines what action is to be taken. A message accom-
panies each menu to help direct and cue the operator's actions. Each button is labeled indicating its associated function.

When input of numerical data is required, a ten digit numeric keyboard appears in the interaction area with additional keys for minus sign and decimal point. The data is entered by sequentially pushing the appropriate buttons with the tablet pen. The accumulated digits are displayed as they are entered by the operator. Buttons are provided to allow the operator to clear the accumulator and begin again, enter the value if all digits are correct, or return with no data entry being made. When the numeric keyboard is no longer necessary in a data entry sequence, it is removed from the screen.

The rightmost lower portion of the display is used for the Width-Height Indicator (WHI) display to be used in conjunction with the normal Az-El display. Vertical and lateral glideslope errors can be plotted for any one of the aircraft shown on the Az-El display. The WHI has a higher resolution scale (±200 ft.) than the Az-El display and is used to determine more accurately the aircraft position close to touchdown. The system automatically selects the aircraft closest to touchdown for display unless this selection is overridden by the operator. The symbol correlated with the index number for the aircraft is used to plot the aircraft position on the WHI.

MULTILEVEL SOFTWARE DEVELOPMENT

Implementation of the display software was conducted in a top-down manner using the chief programmer team management philosophy and the guidelines of structured programming. The system was developed in evolutionary stages with the first stage testing the high risk area of intercomputer communications.
and the central computer's operating system. In the first stage of the display program, skeleton modules ran under a primitive executive. As the implementation progressed, increasingly complex functional aspects of each module were developed.

The most important benefits of this approach were the early visibility of a working program, early assessment of the hierarchical interaction concept, simplified testing and integration with the risk reduction of integration problems, and elimination of the "90% complete" syndrome. Code review and the use of a program librarian were not significantly successful. The program team did achieve the ability to review each other's system designs and description with freedom from insulting each other's egos.

SUMMARY

The approach taken in this display design has led to an operational system which meets the objectives set forth in the introduction. Operators have complete control of the All-Weather Automatic Landing System from a single display console which presents an uncluttered view of the landing situation. The system is easily learned, operated, and requires no knowledge of computer systems. The hierarchical interaction strategy for data entry and system control has been proven to be both accurate and fast. Measurements taken on operator performance have determined that a 0.7 second per button selection rate can be maintained with the hierarchical interaction technique. The operator entry of all initialization data required for an automatic landing sequence (some seventeen virtual button selections) is accomplished in only twelve seconds. The operator is aware at all times of the permissible controls which are active and what possible actions he may take. Favorable comments have been received from a number of experienced landing system controllers, who feel the interaction and display concepts will increase operational safety.

ACKNOWLEDGMENTS

This effort was supported under Contract N00228-75-C-4544 with Mr. John Mulhern of the Naval Electronic Systems Engineering Center, Vallejo, providing many helpful technical suggestions. Discussion with Mr. Richard Witz, NAVELEXSYSMD, Washington, and Mr. Ken Poteen, NESTED, Patuxent River, Maryland, were also valuable during the development of the display system.

Implementation of the computer programs was accomplished by George Firkins, Susan Rubin, Lowell McMahahan, Leo Hoffman and Al Crown. Project support was provided by R. L. Goodman and R. L. Johnson.

REFERENCES

ABSTRACT

This paper examines the potential impact of computer and information technology on the role and mission of the library. In an indirect, but no less significant sense it also considers the impact of the library on the technology. The specific concerns focus on the library as a mechanism for making computer and information technology directly available to the public. The paper also reviews a pilot project being conducted by the New Jersey Institute of Technology involving the Newark and Millburn public libraries in New Jersey.

INTRODUCTION

As of January, 1976 a number of powerful calculators have been installed in the Newark and Millburn Public Libraries as well as the library of the New Jersey Institute of Technology. The Newark Public Library serves an urban community which reflects the diversity of population and the multiplicity of problems facing most major urban areas in the country. In addition, the Newark library is the official state regional library for the northern New Jersey area and as such offers a number of unique collections pertinent to this mission. The Millburn Public Library reflects an upper middle-class suburban community. A majority of the high school students go to college and the working adult population is largely businessmen or professionals. The NJIT library services a scientific and technically oriented university community of about 5,000 students. It is largely a commuter-oriented institution for northern New Jersey and the Metropolitan New York area. A significant proportion of the students are from blue collar families and a very active program for the disadvantaged has attracted a good number of these students into the engineering and technical programs.

None of the above mentioned libraries have had any previous experience with computer or information technology. They have not utilized the marvels of computerized reference services, data bases, automated cataloging, etc. Although a calculator might seem a limited aspect of computer technology, in the environment the introduction of these devices represents a major shift of perspective along with a rethinking of roles, missions, and objectives.

Principally, this equipment is for the explicit use of patrons for any purpose they wish. The primary goal, as stated in public flyer, is making directly available to the public via the public library various computer and information services. This is not to say librarians do not use the machines. In fact, the machines installed are in the five hundred dollar range and considerably more convenient than the machines that existed solely for internal use.

No project of this type exists because of anyone rationale. The members of the project team, users, librarians, sponsors, manufacturers may very well perceive this project differently. The implications are perhaps best understood by viewing a number of alternative rationales that are or could be representative of the parties involved.

SOCIAL CONCERN

Approximately two years ago one of the members of the project team was in a Washington, D.C. department store and happened to observe a salesman driving away some young children who wanted to play with the calculators on display. Evidently he did not perceive them as likely customers. To those of us who have envisioned computer technology as accessible to the public such an occurrence is personally disturbing. However, disturbances of this type have a way of generating reflection.

It seems that for many years the view of many professionals was that the all-encompassing computer 'utility', because of economics of scale, would be the mechanism for bringing computer power to the people. While hardware costs have demonstrated their economics of scale, software and communications will remain a bottleneck for quite sometime. Even while this orientation toward the grand large scale system continues for a vast number of prophets in the field, it has turned out that for the people 'it is the little things that count'.

* Attributable by the authors to Richard Wilcox.
computer and information technology to fall in the hands of the public on any wide-scale basis.

Perhaps the events arising out of the selection process that occurs in a competitive market place are hinting that it is not large scale but the small scale we should be looking at for delivery mechanisms. There have been in the past few years a growing number of such small scale-efforts—Resource One in the San Francisco area for example. The majority of endeavors appear to be the volunteer efforts of concerned professionals and community groups. Unfortunately, because of the nature of these activities many of them lack documentation and very little evaluation is accomplished. While many of the individuals behind some of these programs might scoff, we feel they have suffered from a lack of institutionalization. What is sacrificed, of course, is the opportunity for permanency and wide scale transferability.

With these considerations in mind our choice was to start at the bottom, utilizing a variety of calculators with performance and associated price above the level the average person could afford. It was also a conscious choice to institutionalize the effort—but why the library? In one real sense it is like the explanation of why people climb mountains: they are there. The library is an institution that is available to the public, its personnel are familiar with serving the public, and it is relatively neutral with respect to political, social, ethnic, and organizational polarizations. From the point-of-view of a person who is interested in delivering computer technology to the public the library is the convenient place to do it.

Why one wants to put advanced technology in the hands of the public is probably a question each individual should answer for himself. However, one view is that it may be increasingly difficult to exercise the privileges of an intelligent citizenry in a democratic society without an understanding of the capabilities and limitations of the technology which is beginning to monitor, regulate, and perhaps control aspects of that society. The key issue seems to be: "Will the utilization of this technology by society be such that each citizen must have a right of access and availability in order to function as a part of the society?" Suppose today we took a group within our society and denied them use of the telephone? One could easily list a set of severe consequences for such a group.

LIBRARY MISSIONS

The average person probably equates the usual library with vast arrays of printed material such as books or other physical images like films, records, etc. The more appropriate view for our purpose is to consider the library as an institution for allowing people to utilize information. Utilization implies not only storage and retrieval, but creation, organization, and manipulation as well. The use of the technology to allow patrons to be able to directly perform these latter operations implies a host of information services that have not previously been possible:

- Allow the library to support transient information needs of its user community.
- Provide mechanisms for patrons to exchange information.
- Establish the Library as a Learning Resource Center.

A place where individuals of ages 5 to 85 can go to take advantage of the educational options offered by the technology.

The possibilities stretch from rather straightforward electronic bulletin boards, text processing, discussion systems and CAI implemental on microprocessor oriented intelligent terminals with floppy disks or digital cassettes to more future oriented visions as those expressed eloquently by such individuals as Kagan and Nelson.

In essence the key ingredient is the concept of easily dealing with "variable" information. Information subject to change and modification on a frequent but unpredictable basis.

The choice of the library and its institutional characteristics has led us to another premise in our design of this effort. Most libraries have been reasonably successful in the performance of their mission because the services they provide have no short term dependency on other organizations in the delivery process. Short of the dramatic act of closing a library, most sponsoring institutions do not (at least not yet) instruct libraries to sell books to raise money, whereas colleges can be told to cut down enrollments, offer less courses etc. With this in mind a choice has been made to focus on equipment which is self-sufficient, and not dependent on outside computer power. This is not to say that if we install at some point, an intelligent terminal or word processor, it will not have the dual-use capability of being able to talk to a computer. In fact, any reasonably sized organization buying word processing equipment that cannot be upgraded for this, has not looked to the immediate future.

EDUCATIONAL ASPECTS

Although there are public schools which offer experience with computer technology, they present a
distorted perspective of the extent to which technology is actually being taught to the public. There are many outstanding programs in largely upper middle-class communities or in communities in proximity to a university. These efforts represent only a fraction of the pupil population. For example, in the near future, it is unlikely the Newark New Jersey school system would be able to replicate the six thousand dollars worth of calculator equipment available in the Newark Public Library within each of its schools. Even if it did, the equipment would not then necessarily be available to other specialized educational programs without added staff costs to open school facilities after school-hours.

In the case of a public library, one must consider a diversity of user needs for computer technology that must be met. They run the gamut from advanced research to recreation. The objective of expanding computer technology depends, as well, on the nature of the user population. In the case of this study, equipment for the Newark Public Library (a lower-income area library) may not be appropriate for Millburn (an upper-middle income area library).2 Similarly, one would allow for more sophisticated equipment to be included in the library at NJIT, a technical college library. The sample of this study was especially chosen to represent two socio-economic communities as well as a technical college community. It will thus be possible to evaluate differences between the groups from the point-of-view of the appropriateness of the new equipment. The principal question to be answered is whether this advanced media equipment represents any improvement over past services.

The library has traditionally been the place where individuals go to learn on their own. It serves as an important place where users can pursue their own information needs. Librarians continue their efforts to encourage users to consider learning as a life-long process. Therefore, an objective of this study is to investigate if the library cannot serve as an improved educational agency. In addition to introducing computer technology to the public, this service of the library could be used efficiently in the educational process.5 It is clear that the adult population particularly from thirty on up has not been adequately serviced by traditional educational programs. Many adults no longer have the opportunity to attend school. For those that do, there are many problems associated with returning to college or even to graduate school. Imagine a practicing engineer in a class of engineering students becoming aware that he knows less than some of the young students! Technical areas of instruction, particularly where updating is concerned, are ideally suited to CAI (Computer Assisted Instructional Systems) made available on a self-service basis in libraries. One of the earliest lessons to be learned by those involved in designing operational Management Information Systems is if you want senior managers to use these systems and do not have enough terminals for every office, you put it on wheels so the manager can take it into his office and not publicly demonstrate to younger employees the mistakes he is making in learning the system.

NJIT has developed a number of specialized educational programs to provide opportunities for disadvantaged students to enter technical fields. These programs have partially blurred the distinctions between course materials offered in the high school and in special first year programs at the university. They have entailed tutoring efforts, summer sessions and weekend classes for high school students, special classes for high school teachers, as well as the development of instructional materials for the use of high school teachers.

Educational institutions involved in the programs for the disadvantaged, could pool their resources, and invest in Learning Resource Centers located in libraries. From an educational standpoint the availability of computer technology able to support drill and practice as well as the updating of basic knowledge, would greatly alleviate the coordination difficulties that result from offering remedial services in public schools, universities, and adult programs.

This pilot project will emphasize guidance and instruction in the use of computer technology to teachers. Their willingness to use the media made available is essential to the success of this program. Their perception of the role of educational technology is of the utmost importance. It will determine the length of time they are willing to spend developing specific educational lesson plans for the equipment. They will be encouraged to take advantage of the calculators to design stimulating assignments for children. In mathematics, children could easily handle more challenging problems such as computing a growth curve to fit the daily growth of a classroom avocado plant. This computation would normally be too laborious without the use of a calculator.

From the point-of-view of educational theory, there appears to be a shift from concern with teaching to a concern with learning. Whereas, teaching provides a framework for learning, teacher-centered learning does not develop independence in the learner. Teaching is done to the individual, not by the individual.

Educational technology may be an effective technique for students to acquire skills independent of the teacher.

A question to be explored is what types of individualized learning experiences students can handle in different grade levels as a result of educational technology? Also, computer technology is especially important for providing work at varying rates for students at the extremes (faster or slower) than the average classroom. It is expected the library could serve as an adjunct to the teacher for these students. For example, this study will use drill and practice
techniques as one effort for those needing remedial mathematics. Applying the psychology of learning to practical teaching problems, the student is presented with individualized instruction in arithmetic. Actively involved with his own performance of arithmetic problems, the correct answers via computer print-out serve as an incentive in the learning of repetitive tasks. It is no longer necessary to wait to have a test scored. As far as assisting the classroom, drill takes up considerable time. Drill and practice take time away from teacher-contact with students that could be used in better ways.

Generally schools and libraries have co-existed as separate units. An objective of this study is to determine professional (librarians and teachers) roles in the use of educational technology. It is possible that teachers will be more receptive to educational technology when it is in the library. They may not sense the equipment as a threat to the teaching force.

Many teachers have been dubious regarding the importance of the new material and have not taken it seriously. This study will utilize survey instruments as well as interviews to evaluate the attitudes of teachers and librarians. They will also be working with the media specialist in the selection and utilization of materials. Results from questionnaires will serve as an up-to-date monitoring system to determine which tools are serving advantageously and which are falling short of expected outcomes. The results will be used as a basis for future priorities, deciding the long-range goals of this program.

PROFESSIONAL CONSIDERATIONS

The concept of technology assessment has emerged in recent years as an issue of primary concern to society. Most assessment studies have been executed as paper and pencil efforts. It has yet to be realized that there exists a need for assessment by experimentation. From one view this effort could be considered a 'live' technology assessment. A major goal in this assessment effort is to gain insight into the unexpected consequences of technology. We expect to observe results we could never have predicted. A few unanswered questions in this area are:

1. For what purposes will the technology be used by the public and what are the benefits or limitations?
2. Will the library schools have to train educational librarians (will SLA add a new component to its membership)?
3. Is the technology designed to really meet the requirements of this mission?
4. While we have in Information and Computer Science had some success in designing to deliver "Information," do we know how to design to deliver "Information Technology"?

Even at the calculator level the variability in human interface for machines of similar capabilities is quite significant. In terms of advanced machines, the layout of the keyboard and the associated instruction manuals appear to be either for the sophisticated user or the individual who must learn to use the machine. The developments in this area have been so rapid that few of the companies appear to have made any evaluation efforts in terms of market assessment or product development. This may change as the market saturates.

Another aspect of the assessment issue is the observation that Congress seems to be dragging the National Science Foundation, kicking and screaming, towards a program called 'Science for the Citizen'. No one at this point seems to be sure what that program might encompass. One would hope the delivery of advanced technology would be a key aspect. It is quite clear that organizations such as the National Association for the Public Assessment of Technology would tend to support a view that the public should have a right of access to information and information technology if it is to have an equal voice with those institutions that can afford these benefits.

Technology has moved so rapidly it always appears that implications for the training of professionals in these fields takes place as a reactive process as opposed to an anticipatory one. For example, the vast majority of development and design activity in the professional community represented by ASIS has dealt with the technical means of retrieving data for the user but not the processes of allowing users to create, store, manipulate and update data.

We can foresee that as a result of programs designed to improve the availability of technology for the public, there will be a further blurring of the divisions between Computer Science, Information Science, Library Science and the Social Sciences related to educational processes.

ISSUES AND PROBLEMS

This paper has raised a number of issues and problems. Whether it be at the limited level of the calculators, or the more sophisticated equipment we hope to incorporate later, the following is a summary of issues that must be examined, evaluated and resolved before one can expect any wide scale implementation of programs of this type.

Community acceptance

There are a great many individuals with unimpeachable credentials whose current reaction to computer and information technology is to avoid it like the plague.

One can point to the existence of this antagonistic attitude represented even on our own faculty. Perhaps
more serious a concern is the feeling that the technol­
gy is an influence in widening the gap between the
advantaged and the rest of society. Apparently
some community organizations have taken positions
that they do not want calculators allowed in public
schools because it will degrade the child's learning
of basic math. Also, some experiments with the avail­
ability of CAI systems, have indicated that a child
having difficulty in relating socially to other children
may further withdraw.

This latter phenomenon is also behind some success
stories with Management Information Systems. Given
an organizational environment that has severe restric­
tions on human communication, an MIS system that is
designed to be very reactive at a terminal, can easily
become an unconscious surrogate for human communica­tions—the computer becomes someone who will
listen, obey and respond.

One key element in the library approach is the
volunteer nature of the effort. The use of the calcula­tors is a self-made choice by the individual. This may
lessen possible friction involved if these were imposed
directly into an educational program. We are trying
to maximize the availability of material on games and
puzzles which represent an unconscious form of learn­
ing, and the existence in the program of the 'Computer­
Tutor' and the small 'Quiz Kid' means the teacher can
courage those children who need basic drill and
practice to engage in such. The more advanced ma­chines for either business or scientific use can only be
mastered if the individual knows or attempts to learn
the basic applied math concepts that are presumed in
their design. Also we have undertaken to instruct
those teachers who want to learn as to the types of
lessons that could be assigned in math and science areas
that would not be convenient without these machines—
e.g., curve fitting to plant growth data. Since we are
not set up to have a full-time educational staff avail­
able, the procedures encourage people to help one an­
other to learn. Initial eyeballing of what is taking
place appears to indicate that children are most often
learning and doing on a machine in groups of two or
three rather than one alone. Maybe our concept of a
CAI workstation, with one child working alone at a
terminal, bears re-examination.

CAI design

The CAI area can be characterized today by two
diametrically opposite approaches: One is the large
system with a great many hours of effort put into de­
veloping a single lesson by specialists trained in the
intricacies of the system. This could be represented by
the PLATO system and the TUTOR language at the
University of Illinois. An alternative approach is the
PILOT CAI language which represents ten simple
commands that can be taught in about one hour and
allow any teacher to prepare straightforward lessons
to be used by students. Interestingly, the PILOT has
been implemented on the DATAPoint Intelligent
Terminal, and can be implemented on any of the similar
devices made by such companies as DEC, WANG, etc.
The PILOT language can also be taught to sixth
graders on up and utilized to allow students to design
lessons for other students which represents another
intriguing learning approach, particularly for the stu­
dent designing the lesson.

Basically, these two approaches can be distinguished
by: one, canned lessons prepared by specialists and
two, tailored lessons modified or created by the indi­
vidual teacher and the students. The latter is imple­
mentable on stand-alone devices costing today about
twelve thousand dollars. One can infer from other
parts of this paper that we feel this latter small scale
approach belongs in public libraries as part of a Learn­
ing Resource Center concept.

Some other rather obvious issues we have already
discussed are:

Institutional Cooperation—ultimately the problems
of cooperation of educational and library institutions
have to be faced, not only at the funding and admin­
istrative level, but at the training level of professionals
employed by these institutions as well as the research
and evaluation efforts associated with these endeavors.

Recreation and Learning—it is really not clear that
we understand these processes or their relationships
and potentials within the context of the technology we
have been describing. The little calculator has already
raised issues about possible dis-benefits to the student
without accurate evaluation to support any of a multi­
titude of views. The calculator is only the tip of the ice­
bberg with respect to the potential technology.

Equipment Design—Current equipment design is far
from satisfactory for the purpose of general public use.
This is even more true as we move up the ladder to
intelligent terminal capabilities. Also, instructional
material supplied by manufacturers is usually written
with the assumption the user has to learn the machines.
Security of the equipment is another aspect that has
only received minor attention from some manufactur­
ers.

Problems with Change—Both librarians and educa­
tors can suffer from the same psychological resistance
as other adults reluctant to reorient or engage in up­
dating their knowledge and rethinking basic goals of
their endeavors. As a whole, we have been very
fortunate with the individuals involved in this project.
However it does take time and patience to bring about
an understanding of an endeavor of this type as it is
not established practice for libraries. Also, a number
of libraries that have made available simple hand-held
calculators, in the twenty dollar range, have had to
drop the programs because of maintenance and loss
problems. Most public libraries do not usually have the
background to judge the machines required and to
evaluate their capabilities. It has been our approach
to provide the expertise to utilize advanced equipment

From the collection of the Computer History Museum (www.computerhistory.org)
that is not competitive with what the average person can afford. This then provides the library patron a service that is potentially attractive in terms of what he or she may need to accomplish.

REFERENCES


APPENDIX—SCENARIOS

One way to place the goals of this program into concrete terms is to list some specific scenarios which show the various uses to which this technology located in the library can be put.

A high school student uses a calculator to do some calculations on physics homework.

A housewife uses a calculator to balance the family checking account and evaluate the impact on the planned budget.

A professional (engineer, accountant, etc.) comes to the library to utilize a more powerful calculator than he or she could normally afford.

Many teachers at local schools can restructure their homework assignments in the math and sciences to reflect the availability of calculators at the library.

A secretary and/or treasurer of a local club, organization, community group walks into the library carrying his data file on membership and finances in the form of a small digital storage device—e.g., digital cassette, minitape, floppy disk, etc. He or she goes and sits down at an intelligent terminal that has been set up as a text editing and composing device and performs the updating on recent changes in addresses, membership, dues, expenses, etc. He or she then prints out a summary and duplicates copies for the membership.

A student utilizes this same storage media and terminal to compose and edit term papers over a period of time.

A new form of book called the 'interactive book' has become very popular. With this book stored on a digital storage medium, the reader can play the role of one of the characters so that as he reads he can make choices at strategic moments in the plot which result in different plot resolutions.

Even the librarian can utilize the word processing equipment to prepare various reference lists with continuous updating capability.

A young couple walks into a library and asks for the digital cassette holding the advertisements of babysitters available in the neighborhood: a teenager walks in to add his name and description to the same cassette.

A teenager requests the cassette or floppy disc advertising part-time help available for cleaning, gardening, snow clearing, etc., so he can add his name and capabilities to it.

A citizen requests the cassette devoted to discussing a new and provocative book so he may review what other people have said and add some of his own comments.

A teenager requests the cassette of an anonymous discussion of dating problems by teenagers. An adult requests the cassette dealing with marital problems.

A group of local businessmen or procurement officers have a cassette on which they are exchanging notes on the performance of various supplies, equipment and suppliers.

A cassette is being used by people in the community to discuss their views on various local issues, say a rezoning item. A copy is sent once a week to the government group involved.

Various cassettes have been prepared and are maintained by various local government and community groups to provide information on services, volunteer work available or needed, etc.

A psychologist is utilizing cassettes in the library as a form of random group therapy sessions.

A social worker has organized and monitors a number of problem oriented conferences among people having similar difficulties.

A class discussion cassette has been established with a
local teacher to augment the regular class hours and provide more opportunity for free discussion. This is a particularly popular facility for use in adult education courses.

A group of stamp collectors utilize a cassette to bid on stamp trades.

A group of professionals in a common area of endeavor utilize a cassette to exchange information on recent papers and findings in their field.

A teacher delivers to the librarian a floppy disk containing a CAI lesson she has prepared for use by her class.
The expanding role of on-line interactive searching

by VIVIAN S. SESSIONS

The City University of New York
New York, NY

ABSTRACT

On-line interactive searching represents the newest thrust in information retrieval. The combined technologies make available to librarians and other information professionals large volumes of bibliographic data which reside in computers as remote as thousands of miles from the terminals using them. Presently the individual data bases that furnish the examples of actual usage (ERIC, NTIS, Psych Abstracts, etc.), are accessed primarily via Lockheed, SDC, Medline, and the SUNY Bio-Medical Network. Although it is difficult to give a definitive statement on the proportionate economic contribution of this kind of information retrieval to the entire computer economy, estimates of numbers of characters of information on-line (25-50 billions), indicate a significant dollar involvement. It is in this framework that bibliographically-oriented members of the computer community urge their non-bibliographic colleagues to understand information retrieval and to heed its requirements for further development.

Of all the library and information center usages of computers, the one that concerns this paper is on-line interactive information retrieval. It is retrieval in the sense of finding references to documents relevant to a particular problem in data files that are comprised of bibliographic records. Not all such data files can be used either interactively and/or on line, but those that do not qualify on both counts are excluded from this discussion in order for us to zero in on the I.R. mode that is today most visible in terms of public discussion, in terms of growth, and in terms of challenges to the computing community.

A typical form of on-line interactive retrieval would be the situation in which an information professional uses a terminal with dial-up capabilities to log into, say, Lockheed Information Systems in California via a communications network. S/he indicates a particular data base (the Educational Resources Information Center (ERIC) is a good although over-used example), and then proceeds to process such a query as the one recently done at the Center for the Advancement of Library-Information Science at the City University of New York. This particular query was for references to materials on collective bargaining in higher education. The CRT quickly indicated that the particular way in which the query was asked would yield 367 references in the ERIC data base. The terminal operator asked to have a few displayed, and when all of them seemed on target to the requestor who was sitting by, a message was sent to have the entire number printed out on the high speed printer in California. In a few days the information arrived in New York.

As this was one of the less expensive data bases, and very little computer time was needed to develop a strategy, the total bill was about $40 for the combined use of the data base, the communications network, and the high speed printer—mostly the latter, with 367 citations @ 10¢ per citation printed. This cost excludes, of course, the original investment in the terminal and the time of the information scientist who served as the interface between the requestor and the system. Since the requestor was satisfied with 367 references, there was no attempt to query the file with alternative strategies, thereby keeping the connect time, the most important cost item, at a minimum.

Characteristic of the kinds of systems under discussion in this paper is the fact that the data bases are loaded on computers that are remote from the terminals accessing them—remote not just in the sense of being in a different part of the same building as the terminals, but remote as measured in hundreds or thousands of miles, and housed in different organizations. Even the creators of the data bases (The National Technical Information Service, American Psychological Association, to name a few), once they send their regularly scheduled up-dates to Lockheed or SDC, then access their own files over the same kinds of terminals and via the same kinds of communications networks as the users of their data bases. (Some specialized exceptions do not affect the general situations.)

Remote on-line information retrieval only became a publicly available resource in libraries and information centers in 1972, with a limited number of files. By 1974 Lockheed and SDC had developed their re-
spective capabilities to the point that the Information Industry Association awarded them jointly the Product-of-the-Year award in 1975. In the year since the award they have made considerable advances in the terms of services and customers. Together these two private corporations now account for two-thirds of the kinds of bibliographic data now available for on-line remote searching in the U.S., the other third being provided by public systems, the files of the National Library of Medicine and those of the SUNY Bio-Medical Network. Although there are large overlaps in the data bases handled by these four systems, they can be considered as being independent contributors to the computer economy, for reasons that will be pointed out shortly.

The fact that some of the Lockheed and SDC data bases are government-created and some are private in origin, such as the Institute for Scientific Information (ISI), is of significance to the user only inasmuch as the privately created data bases are more expensive; methodologies for using them differ to the extent that every data base has its own characteristics as to coverage, access points, and hard copy aids to users. And all these are constantly changing.

In the decade before the remote on-line interactive developments, many librarians were already involved in information retrieval. If they worked for organizations with the resources necessary to mount their own bibliographic information systems, they could directly participate in the vanguard of their profession. However, these opportunities were limited to librarians in large corporations and government agencies; the public and university librarians were out of it except for those few places that had contact with information dissemination centers.

Now, however, direct participation in the information retrieval process is possible to anybody having access to a terminal with dial-up capabilities, access to a purchase order of a few thousand dollars, and access to some training (more is obviously preferable). Although many information professionals may be forced by institutional reasons or personal inclination to continue depending on intermediate search centers, the desire to be independently on-line to immense store-houses of bibliographic data is a matter of wishful thinking and more a matter of persuading internal management of the desirability of such action. The advantages to the individual information professional for upgrading his/her skills aside, the advantages to the organization are enormous. The more its own professional information staff knows about information retrieval, the more that organization is going to profit from it.

"On-line" and "interactive" are usually stressed rather than the "remoteness" of the data bases in present-day discussions of information retrieval. All three are intertwined. It is the on-line capability that brings remote access to life, and it is the interactive characteristic that really separates on-line from batch searching. Without the interaction, information retrieval on-line offers little advantage in search strategy to the batch mode.

The interaction now possible on the standard on-line information retrieval system is not as conversational as the term implies, but has divided itself into systems—type interactions (such as the user being told that the host system is on line, or that the operator has made a syntactical error), and interaction wherein the user knows immediately (computer-time immediately) how many document references would be retrieved in the NTIS system by using the search terms being investigated. For instance, how many "hits" by using the search term "service industries", how many would be found by the term "innovation", and how many by the combination of the two in an AND strategy? It quickly developed in this search that the results would be insufficient unless "service industries" was broadened to include "services" in general in an OR relationship.

To illustrate another interactive feature in the very versatile NTIS file, the information professional looking for material on "information theory" might do well to use the EXPAND command, which would then put on the CRT all the descriptors in which "information theory" is only the first part of a multi-word descriptor. This could result in a decision to either broaden or narrow the search, the latter being done by asking only for "information theory, hand widths", if that is more appropriate. The EXPAND capability is particularly important in systems that either have no printed thesauri, or in situations in which the user has no immediate access to the thesaurus.

One of the most powerful features of interaction has already been mentioned in connection with the ERIC search earlier: the ability to display a few citations to see if they are on target, thereby enabling the searcher to decide whether to continue modifying the strategy, or to work with what has been retrieved, and order a print-out. If the searcher is using a terminal with a printer attachment, or a printing terminal, the only decision left is whether the results are either sufficiently short or sufficiently urgent to have them printed out on the spot, or to leave them to the high-speed printer and the mail. These few examples of the effects of interaction on search strategy merely scratch the surface.

The intricacies already mentioned of searching interactively, and many other fine points extending the possibilities, are starting to be taught to the librarians or other information professionals in a variety of formats. This is quite different from library automation, which concerns itself with circulation control, catalogue card production (now handled in volume through the Ohio College Library Center) and related administrative tasks. While such uses of the computer con-
tribute heavily both to the smooth functioning of the library and to the computer economy, it is information retrieval which responds to the expectations of the information user community.

Although the ERIC search on collective bargaining mentioned as an example seems to meet those expectations completely, it worked out so well only because of specific conditions that existed. All the tricks of computer technology would have been to no avail if there had not existed a data file in which there was a high probability of the information existing, if the information professional did not know of the existence of the file and how best to access it, if she had not had a purchase agreement with Lockheed and an appropriate terminal at her command.

If these facts seem elementary, it will surprise the computer community to report that great numbers of people with some knowledge of computing call our office without either knowing or being willing to learn these basic facts. And these callers are not limited to those whose knowledge of computing is purely peripheral. A gamut of callers assume that because they have access to the University’s central computer facility they therefore have access to either ERIC or the other files on the Lockheed, SDC, Battelle, Medline, N.Y. Times, or other systems. It is an unpleasant job to convince them that their research allocation of $400 will not contribute one dollar to the cost of using any of the bibliographic data bases that are available in the interactive, on line mode that they are starting to hear so much about.

This type of problem occurs most often with seekers of information who are neither library nor computer professionals, although even with those groups some confusion exists. The real problem with the computer professionals, however, is that although they understand the concept of files and differing conditions through which they may be accessed, they often have little understanding of how bibliographic files are created; this makes it difficult to carry on a meaningful discussion on how to use them for information retrieval, whether on-line or in batch. Too many graduate computer scientists confuse the experimental work being done in automatic content analysis with the real world in which the analysis is done by human beings, and where data bases are created using descriptors, or subject terms, from a controlled vocabulary usually called a thesaurus.

And everybody is confused by the constant change in what is possible. Each data base has its own characteristics, which influence the way it can be accessed either at all or most effectively—however these characteristics change so rapidly that a file which one day can only be accessed by descriptor may, shortly after, be searchable for any word in the abstract (COMPEDEX is the most recent example). Also, new data bases are being added so frequently that a query which a few weeks ago might have been relevant to only one data base should now be considered from the point of view of, say, Dissertation Abstracts.

It is no more reasonable for me to expect computer scientists to keep up with every change in information retrieval than it would be for you to expect me to be completely au courant in the hardware. What I do ask in this area is that the computer professional be sufficiently educated about the conditions of on-line interactive retrieval so that he can play a positive role in the further development of this important sector of the computer economy. The exact nature of the role may vary all the way from not being a hindrance to affirmatively helping solve some of the problems in computing that affect many usages besides I.R.

The “not a hindrance” part of this request has two categories, the first of which is not to play pro in I.R. if your computer expertise is in other areas and, two, not to downgrade I.R. by giving it low priorities in terminal availability, budget for external computing, and training opportunities. Information professionals involved in on-line retrieval deserve terminals that are at least the equal of those assigned to the programmers in the organization. They have enough problems dealing with the intellectual problem of I.R. without in addition having to fight the equipment.

Both the “no hindrance” and the much more active involvement in on-line I.R. to be mentioned shortly are really problems that transcend a particular computer installation, and therefore belong in the domain of the entire computer economy as represented in the Conference. Anyone manager has his/her own budget to optimize, his/her organizational hierarchy which determines priorities apart from any personal interest in I.R., and assorted additional internal computing problems that take immediate precedence over an operation that is on-line to an external system. The economic contributions of I.R. in any mode, and especially in the on-line interactive mode, are to the industry as a whole, and that is the reason for their being presented in a conference that represents the industry as a whole.

It would be easier to make the point if it were possible to make some claims for I.R. in terms of its proportionate value to the industry. What proportion of CPU time in the country is devoted to I.R.? What proportion of equipment costs, mainframe and peripherals? What is the proportionate use of programmers systems, analysts, data communications experts? Unfortunately, these questions are impossible to answer, even if put in absolute rather than proportionate figures.

Trying to arrive at some other measures for evaluating the role of on-line interactive information retrieval in the computer economy, some interesting estimates of absolute figures did emerge. The Lockheed Information System has eight billion characters of information on-line 14 hours a day (they start before dawn New
York time in order to accommodate European users). SDC contributes a like number of characters of information, and the bio-medical information systems (the National Library of Medicine and the SUNY Bio-Medical Network) a little over another eight billion together. These four systems are, therefore, making between 24 and 25 billion characters of information available currently in the interactive mode, most of them being used in the U.S. in locations remote to the residency of the data bases. Martha Williams of ASIDIC estimated recently that one million searches a year were being done on those four systems.

Since the figures as to sizes of the on-line systems were estimates of current availability obtained by informal telephone calls to appropriate people rather than from official reports, the same informality (combined with other conversations) persuades this writer that by the time this paper is actually presented, closer to 50 billion characters will be available for on-line interactive searching in the U.S.—especially if one takes into account the information systems available through Battelle, The Information Bank (N.Y. Times), the various ones just coming up, and the normal growth in the size of files already in existence, as well as the expansion in amount of information in those files that can be searched interactively. There are also many international systems (the International Labour Organization, the World Health Organization, the U.N. Environment Centre, etc.) that seem on the horizon for direct access by U.S. information professionals, that will either contribute to the fifty billion figure or enlarge it.

The point to using characters of information as a measure of size is first its availability, and second its meaning to the computing community in what this means to the economy in terms of systems and programming support, hardware investment, and CPU utilization.

This approach does not solve the problem of how large information retrieval is proportionately to all other uses of the computer, including those for library administration, but it does indicate a contribution sufficiently large to make the point that the needs of information retrieval should be seriously considered in future developments of the industry. The size of information retrieval should also spur all information professionals, whether library trained or with other backgrounds, into demanding more than mere “no hindrance” policies mentioned earlier.

The most obvious need is for more education, education in which the computer professionals and the information professionals cooperate. Even with the development of such courses as those that have been offered in the City University’s Professional Development Program for Library-Information Science, too few people have faced the problem of really bridging the gap between computer science and its bibliographic information systems implementation. Courses in programming only begin to bridge the gap; they must be supplemented by courses in how to measure the effectiveness of various kinds of programming. In the on-line remote environment, data communications can scarcely be ignored or confined to sociological discussion. The examples can be multiplied; the point is that educational efforts must be made by all segments represented by AFIPS because they are all involved in information retrieval.

A second “must” lies in the area of data storage. The closets are becoming filled, and will be more so when there are significant breakthroughs in input methods. The possible argument that information personnel already has at its command the very large amounts of bibliographic data already mentioned is not valid, if one starts to point out the large gaps in the bibliographic coverage of the literature, particularly in the social sciences. The systems need large capacity, and the librarians will have to learn to work with those larger capacities.

A third “must” lies in the area of referral vs. bibliographic services. There is too large a duplication of the latter—the overlap in data bases between SDC and Lockheed, for example, may be anti-monopolistic, but it uses capacity that might better service society’s total communication needs by the provision of more referrals from file to file. As the number of files on-line grows, so will the necessity to refer even the expert to the correct file, and to instruct him/her properly in how to convert a question from one system to another (assuming that total file compatibility is an ideal rather than a complete reality).

A fourth “must”, at least to this writer, is for a more imaginative use of terminals in information retrieval. We now have two possible ways of using the terminal on-line: letting the information roll off the screen or, if available, transferring it to a printing device. Some exploration might well be made of the divided screen, so that information (perhaps on strategy) can be kept in view while the materials to be scanned are rolling off the screen. The technology is presently used in another type of information system, and its application to I.R. use merits some study. Perhaps several screens, with or without divisions, will be needed to make full use of large volumes of material being on-line at once.

The topic of this session: “Enhancing Library Services Through Computer Technology” could have been interpreted as supplying the specifics for on-line interactive searching. However, those details are findable to the kinds of librarians and information center professionals who are likely to attend this meeting. This occasion should be used instead to broaden the strokes, to enlarge the possibilities, to engage the cooperation of the entire industry. It is my earnest hope that this has happened.
SCIENCE AND TECHNOLOGY

Computer and Data Base Architecture
Software
Computer Science
Applications of Computer Science
Developing application oriented computer architectures
on general purpose microprogrammable machines

by TOMLINSON GENE RAUSCHER
NCR
Cambridge, Ohio

and

ASHOK KUMAR AGRAWALA
University of Maryland
College Park, Maryland

ABSTRACT
Surveying contemporary commercially available computers reveals a general incongruity between computer architectures and the problems the computers are being used to solve. Surveying the commercial applications of microprogramming reveals that microprogramming remains largely an alternative technique for manufacturer implementation of basic machine language instruction sets. With the large number of contemporary general purpose microprogrammable computers (especially minicomputers), the advantages of microprogramming are available to ordinary users for solving specific problems. From a pragmatic view, the architecture of a computer is defined by the microprograms resident in its control store. Changing the microprograms in a computer’s control store therefore redefines its architecture. Architectures may be defined for specific problems by changing the microprogram in control store for each problem. As problems are represented by higher level language programs, compilers can automatically generate a microprogram for each higher level language program. The generated microprogram, when loaded into control store prior to program execution, defines an architecture that efficiently supports program characteristics. The advantage of this scheme is that it utilizes the power of microprogramming for each user’s specific problem without forcing the user to comprehend the implementation complexities of a particular microprogrammable machine. This paper investigates several techniques for architecture redefinition via microprogramming.

INTRODUCTION
Motivation

Problem solving with the assistance of a modern general purpose digital computer generally involves a sequence of processes.

(1) The programmer writes a program to effect his algorithm in some higher level programming language.
(2) The analysis phase of a compiler for the programming language transforms the program into some intermediate language representation.
(3) The synthesis phase of the compiler transforms the intermediate representation into a machine language program.
(4) A microprogrammed interpreter (i.e., an emulator) directs the interpretation of the machine language program by the machine hardware to produce results.

This problem solving process may be considered to be a series of transformations on various representations of an algorithm to solve a particular problem. Most of the representations have been studied in some detail, and as a result there are well-known algorithm representations, higher level languages, and intermediate language representations. Similarly, most of the transformation processes have been studied in some detail, and as a result there are well-known programming techniques, formal language analysis techniques, semantic analysis and code generation techniques, and hardware transformation units. In contrast, the design of machine language instruction sets and the microinstructions that interpret them have been given relatively little consideration.

Some things have not altered, or only slightly, so, like the Model T, they have remained invariant, upright, slow, inefficient and immune to the winds of progressive technological improvement. I am referring to the basic form and rigid format of our instruction sets.

The instruction sets of many contemporary computers were designed with hardware realization as the foremost constraint. Little attention has been given to the types of operations the computers will perform. As a
result, machine language representation of program constructs are often awkward, and the efficiency of machine code for programs is seldom high. "The majority of the... required work is involved with placating the computer, while a relatively small portion of the work is actually applied to the job." In this regard it is interesting to note that... most of the instructions executed, even in scientific applications, are used for "housekeeping" details."

Microprogramming was formulated more than twenty years ago as a systematic alternative to the usual somewhat ad hoc procedure for designing the control section of a digital computer. Although microprogramming has been used commercially for more than ten years, it generally remains a technique for the implementation of basic machine language instruction sets. Recent developments, such as the availability of fast writable control stores, have sparked investigation into other applications, however, these are mainly special purpose and not immediately generalizable.

The advantages of microprogramming have been mainly engineering aspects: low cost, flexibility, and ease of development and maintenance. Because microprograms interpret instruction sets, microprogramming can be applied to general purpose problem solving to overcome inefficient machine languages.

Environment

Problems to be solved will be represented in procedure oriented higher level programming languages. For the higher level language there is a compiler consisting of two parts:

(1) an analysis phase which transforms the higher level language program into an intermediate language representation and,

(2) a synthesis phase which transforms the intermediate language representation into machine language.

Since a majority of optimization techniques are machine independent, the first compilation phase performs all the machine independent optimizations (such as folding, eliminating redundant operations, moving invariant operations, reducing strength of operations, and eliminating dead variables and assignment operations) before it produces the final intermediate representation of the higher level language program. A general purpose dynamically microprogrammable computer interprets the compiler generated machine code to produce results. "Dynamically microprogrammable" means that the computer has a fast writable control store (much faster than main memory) that can be loaded under program control. Once compiled and debugged, programs will be executed repeatedly. Figure 1 summarizes the problem environment.

Problem definition

Basically, the problem is that computers understand, i.e., perform, machine oriented instructions, and as a result problems expressed in higher level languages must be transformed into machine language programs. The resulting machine language representations of programs are almost always inefficient, and hence the process of computer assisted problem solving in the environment discussed is inefficient. The genesis of this problem has been observed in the constraint of contemporary computer instruction sets to machine oriented operations rather than problem oriented operations:

When engineers begin to seek more efficient encodings for commonly used sequences of instructions, progress toward the modern computer may begin.

It has been observed that contemporary machines use inefficient machine languages to represent algo-
rithms. More generally, the architectures of computers do not efficiently support programs being run. By computer architecture we mean the attributes of a computer as seen by the programmer, i.e., the conceptual structure and the functional behavior. The conceptual structure comprises such considerations as the memory hierarchy (registers, main memory, external store, etc. and their logical connections), the representation of data and instruction sets, and addressing schemes. The functional behavior is determined by the machine language instruction set, the instruction interpretation process, and the instruction execution process. The architecture of a computer is to be distinguished from its implementation which includes the entire set of machine registers (including the various memories), the physical connections among the machine registers, the mapping of the conceptual structure onto the physical structure, and the implementation of machine language instructions. The efficiency of solving a particular problem on a computer depends primarily on the degree to which the architecture of the computer supports the problem primitives. The primitives of major concern in expressing a problem are the data structures inherent in the problem, the operations (transformations) that manipulate the data structures, and the flow of control or sequencing among the operations. These primitives are employed in the formation of a program which represents an algorithm to solve the problem. Since there is a wide range of problems and algorithms (programs) for solving them, each problem would be handled most efficiently by a computer whose architecture supports the primitives of the algorithm that solves the problem. To realize a variety of algorithms efficiently, it is therefore necessary to define architectures for the different problems.

With a given control store size, the factors of concern that measure the performance of a machine in executing a program are the time required to execute the program and the main memory space required to store the program. While time and space are also functions of the machine realization (e.g., the technology from which the memories are fabricated), here we are interested only in the architectural changes that affect the time and space required to execute a program.

The problem under study may be summarized as follows:

Given a higher level language program and a dynamically microprogrammable computer with a fixed size control store, design an architecture (including the instruction set into which the program can be translated) that (when interpreted by the machine) minimizes the execution time and main memory requirements for the program.

**Related work**

A design approach that is presently being popularized commercially is language directed computer architecture. As early as 1967, McKeeman noted the absurdity of expecting modern digital computers (which are designed to be very fast automatic desk calculators) to be suitable hosts for supporting such diverse applications as operating systems and compilers. In the Burroughs B1700 computer, there is a novel "S-Language" for each higher level language. The S-Languages are interpreted by microprograms that reside in main memory or in a faster writable control store. While language directed computer architecture shows considerable improvement over the traditional single machine language architecture, there is still considerable disparity between language directed instruction sets and ideal instruction sets for programs.

Microprogramming has been used to improve the performance of programs in several application areas. Inserting microprograms into control store to support a class of problems results in application or environment oriented architectures. The success of this approach depends on the size of the application area. Good results may be obtained for small, well defined application areas.

The approach developed subsequently is an extension of the language directed and environment directed approaches that does not rely on characteristics of particular languages or characteristics of particular environments.

**APPROACH**

*Implementing problem oriented architectures via microprogramming*

Within the given environment there are alternative approaches to implementing performance improvement by redefining computer architecture for individual problems. Of concern are the flexibility of the implementation and the binding time of the redefinition.

Since it is necessary to define an architecture for each problem, one approach to implementing architecture definition is to design for each problem a special purpose machine based on the structure of the algorithm and its intrinsic data structures. This approach has been used by Shays in designing machines whose algorithms exhibit natural parallelism. Such an implementation, however, yields a special purpose machine for each class of similarly structured problems. The expense in building a special purpose machine can generally be justified only in dedicated situations, such as real time radar signal processing.

From a pragmatic view, the architecture of a general purpose computer (especially the machine language instruction set) is defined by microprograms resident in the control store and the associated implementation which the microprograms control. The microprograms interpret the machine language instruction set and thus define the instruction format, the instruction set, the instruction interpretation process, and the addressing
schemes. Since microprograms control the referencing of machine registers, the microprograms also determine the conceptual structure of a computer's architecture. Changing the microprograms in the control store of a machine therefore redefines the architecture of a computer. Since the problem environment under consideration specified dynamically microprogrammable computers (to reflect capabilities of a large number of contemporary machines), changing the microprogram is a simple task.

Dynamic redefinition

Defining architectures for specific problems requires changing the microprogram in control store for each problem. As problems are represented by higher level language programs that are analyzed by compilers, it is natural to have the compiler generate microprograms.

The approach then is to replace the code generation phase of a compiler by a procedure that performs the following operations:

1. generates a microprogram that defines an architecture for efficiently supporting the higher level language program being compiled (this includes interpreting the machine language programs of the architecture) and
2. generates the machine language program for the defined architecture that represents the higher level language program being compiled.

To execute the compiled program, the generated microprogram and machine language program are dynamically loaded into control store and main memory to redefine the architecture of the general purpose host machine. This approach is called dynamic problem oriented redefinition of computer architecture via microprogramming.

Because control store is such a limited resource, translating each higher level language program directly into microcode is not feasible; there would be too many microinstructions for the control store to hold. The compiler, therefore, must still generate a "machine language" program that, when interpreted, effects the algorithm. This means that there is an almost infinite variety of "machine languages".

Effect of control store availability

Since control store is a limited resource, the performance of the procedure that generates microprograms depends on the amount of available control store. Usually control store is small compared to main memory, so the number of generated microinstructions cannot be large. This means that the number of special instructions is small and that most program operations are represented by instructions in a basic machine language.

The instruction sets of the large majority of contemporary computers are redundant in that many operations can be implemented by several different sequences of machine language instructions. As an early example of redundancy in instruction sets, one research group found that "from the programming point of view, little flexibility would be lost if the set of instructions on the CDC-3600 were reduced to less than \( \frac{1}{2} \) or \( \frac{1}{4} \) of the instruction options now available." Similar remarks apply to almost all contemporary computers.

Since machine language instruction sets are so redundant, an obvious way to obtain more control store for use in architecture redefinition is to remove the microprograms that interpret superfluous machine instructions. A machine instruction is superfluous if none of the operations in the program need be implemented using that instruction. While this method can provide more control store, it has disadvantages. Upon removal of some machine language instructions, the instruction set, though still complete, may be very inefficient. That is, it may require several machine language instructions to implement an operation that formerly required just a few machine language instructions. The instruction set may not be well rounded, i.e., some operations could be easily implemented while others would be implemented inefficiently.

A better approach would be to design a "kernel" machine language which is simple, complete, balanced, and does not have trivial redundancy. Operations in the kernel machine language should include operations for flow of control constructs, arithmetic and logical operations to manipulate the primitive data types, I/O operations, etc. The actual language of course depends on the environment in which programs will run, e.g., for simple business applications there is little need for data types like real and complex or for data structures like sets and lists. Even within a single computer system it is feasible to have a variety of kernel languages for application to a variety of problem types. The microprogrammed implementation of kernel machine language instruction sets uses only a small portion of control store. It is not minimal but reasonable in its implementation of program operations.

METHODS FOR DEFINING NEW INSTRUCTION SETS

Instruction sequence method

Many researchers have observed that instructions in the static representation of a program fall into natural sequences.\(^2\) That is, having seen an instruction at a certain location in the program, some instructions are more likely to follow it than other instructions. Contemporary machine languages generally do not take advantage of this dependence. When considering particular programs to be run on a computer, the obser-
vation about dependence between pairs of machine language instructions can be extended to operations in intermediate language programs. It is characteristic of programs to use the same sequence of operations at many points. This observation is the basis for the sequence method of defining new “machine language” instructions.

The sequence method analyzes the intermediate language representation of a program to find all sequences of instructions and the number of times each sequence appears. The method then compares the microprogrammed implementation of each sequence to its machine language implementation. The microprogrammed implementation of a sequence will show improvement over the machine language implementation because it represents a sequence by a single operation code instead of many operation codes. The sequences that yield the most savings are selected as new “machine language” instructions.

As a result of developing new instructions that replace several instructions, the sequence method will also reduce program execution time. The improvement does not usually approach optimality because the method does not take into account the interaction of sequences, especially the dependence between sequences and their subsequences. As the sequence length increases, the number of occurrences of different sequences decreases. Thus there is a tradeoff in implementing sequences of different lengths as new instructions. While representing each occurrence of a long sequence by a new instruction reduces memory size (and execution time) more than replacing its constituent short sequences does, it may be advantageous to use the available control store to interpret instructions that represent short sequences because they appear more frequently in the intermediate language program.

**Program structure method.**

The intermediate language representation of a higher level language program has been used as input to the architecture redefinition process. This static description of the program does not directly provide information about the run time behavior of the program. To minimize execution time, however, the architecture redefinition process requires knowledge of the run time behavior.

One way of learning about the run time behavior of a program is to execute it many times with several representative data sets. By monitoring these executions, information about the run time behavior may be gathered for analysis. An alternative method is to estimate the dynamic behavior using knowledge of input data and statistical techniques. The program structure method for defining new instructions assumes knowledge about the execution behavior of a program. More specifically it assumes knowledge of the frequency of execution of program blocks, i.e., sequences of operations that are executed sequentially with no branching or selection. (Thus if the first operation in a block is executed, all the operations will be executed.)

For each block, the program structure method compares the execution times of the microprogrammed implementation and the machine language implementation. The microprogrammed implementation will require less execution time than the machine language implementation because few instructions need to be fetched and decoded. The method then multiplies the difference by the number of times the block will be executed.

The blocks that yield the most total execution time improvement are selected as new “machine language” instructions.

Some empirical evidence has been reported to support the efficacy of this method. In one study, Knuth “found that less than four percent of a program generally accounts for more than half of its running time.” Given that forty percent of the time involved in performing machine language instructions is devoted to fetching and decoding the instructions, Knuth’s findings imply that this architecture redefinition method can easily reduce execution time by twenty percent.

**A combined method for architecture redefinition.**

The two methods presented for defining applications oriented architectures may be criticized for using only some of the available information. The sequence approach defines instructions that appear globally throughout a program, however, it does not consider program behavior at execution time. The structure approach defines operations that are local to a part of the program, however, it does not consider the operations that constitute the blocks throughout the program. For these reasons we consider a combined approach.

As in the sequence method, the combined method first finds the different instruction sequences, the places they appear in the program, and the differences between the execution times of the microprogrammed implementation and the machine language implementation. For each sequence, the combined method then multiplies this savings by the expected frequency of execution of the sequence. For all occurrences of a sequence throughout a program, these products (savings times execution frequency) are added. The resulting sums represent for each sequence the total execution time saving when the program is run. Those sequences with the highest savings can then be microprogrammed as new “machine language” instructions.

Defining new architectures in this way reduces program execution time for two reasons. First, fewer “machine language” instructions need to be fetched from main memory and decoded because several oper-
ations have been represented by a single new operation code. Second, the microprograms that interpret the new instructions may be optimized. The amount of space required to store the program will also be reduced because several sequences of machine language instructions are replaced by single new instructions.

EXAMPLE IMPLEMENTATION

Environment

An example implementation has been developed to demonstrate the practicability of problem oriented redefinition of computer architecture by microprogramming. The implementation consists of:

1. the analysis phase of a compiler for a higher level structured programming language,
2. a procedure that takes the output of the analysis phase and generates microprograms that define an architecture to support the higher level language program being compiled and also generates machine language instructions to represent the higher level language program, and
3. a simulator that interprets the microinstructions of the host machine.

The higher level language chosen for the implementation was ULP, a language that is simple in structure and general in scope, yet maintains the philosophy of structured programming. The simulator was developed for the Digital Equipment Corporation PDP-11/05 (See Figure 2) with some minor differences. The most important difference was the assumption of a writable rather than a read only control store. This machine was chosen because good documentation was available on the microinstruction format and the machine language emulator, the microprogram level architecture was fairly simple and general (if somewhat inefficient for general purpose use), and the machine size epitomizes situations where architecture redefinition would be beneficial. The simulator simulates microinstructions that interpret machine language instructions. The architecture redefinition program takes the intermediate language representation of a higher level language program generated by the ULP compiler and performs the combined procedure using a kernel language as described previously.

With the ULP language and compiler, the architecture redefinition program, and the PDP-11/05 simulator it is easy to compare the performance of a program as executed on the PDP-11 architecture to the performance of the program as executed on the redefined architecture.

Simulation results

For the simulations, two ULP programs were selected. The first program is a part of a programming system developed to design telephone exchanges, and the second program computes operator precedence relations for an input grammar. Each program was simulated nine times on each of two different input data sets, for a total of thirty-six simulation runs. Of the nine simulations for a program on one set of data, the first represented the standard PDP-II architecture interpreted by the PDP-11/05 emulator. The remaining eight simulations represented the kernel and redefined architectures (generated by the architecture redefinition program) using control store sizes ranging from 64 to 512 words in increments of 64 words. Each simulation recorded the execution time, the number of machine language instructions executed, and the number of microinstructions executed.

As illustrated in Figure 3, execution time for the simulated programs decreased as the amount of available control store increased. Especially important are the decreases in execution time of the redefined architecture using 256 control store words compared to the PDP-11 architecture (which also uses 256 words of control store). The execution time improvement of more than twenty-five percent may be somewhat modest because small data sets were used to keep simulation times reasonable (less than ten minutes on a UNIVAC 1108). In retrospect, larger (and more typical) data sets would have resulted in the programs spending more of their time in frequently executed program loops. As these loops contained new instructions, significant additional time savings would accrue.

Figure 4 illustrates the effect of the new instruction set on program execution. It shows that the number of executed "machine language" instructions on the new architecture was less than half of the number of machine language instructions executed by the PDP-11.
The defined architectures were indeed oriented toward the particular applications.

SUMMARY

The purpose of this paper was to investigate techniques involved in defining architectures to support different problems solved on general purpose computers. In the selected approach microprograms are generated by a compiler and loaded into the control store of a computer thus redefining the machine architecture for the particular problem program.

The first technique for designing new instruction sets was motivated by the observation that instruction executions exhibit repeated sequences of operations. The sequence approach searches through the intermediate language representation of a program and represents commonly occurring sequences as new machine instructions. The second technique was motivated by the observation that different program parts are executed with different frequencies. From the dynamic behavior, the program structure approach represents the most frequently occurring program blocks as new instructions. The architecture redefinition procedure developed to solve the problem combined the techniques used in the sequence and program structure methods.

An example implementation using this procedure was developed. Simulations provided empirical verification of the procedure in improving performance by defining architecture oriented toward particular applications.

REFERENCES


