U.S. Department
of Commerce
National Bureau
of Standards

Computer Science and Technology

NBS Special Publication 500-83

Proceedings of the Computer Performance Evaluation Users Group 17th Meeting

CPEUG81

"Increasing Organizational Productivity"

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
[2]Some divisions within the center are located at Boulder, CO 80303.

U.S. Department
of Commerce
National Bureau
of Standards

Computer Science and Technology

NBS Special Publication 500-83

Proceedings of the Computer Performance Evaluation Users Group (CPEUG) 17th Meeting

San Antonio, Texas
November 16-19, 1981

Proceedings Editor
Terry W. Potter

Conference Host
Headquarters, Air Training Command
Department of the Air Force

Sponsored by
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued November 1981

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-83
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-83, 320 pages (Nov. 1981)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 81-600155

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1981

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402
Price $9.00
(Add 25 percent for other than U.S. mailing)

CPEUG81

Foreword

The theme of CPEUG addresses the principal challenge of the 80's -- INCREASING ORGANIZATIONAL PRODUCTIVITY--as well as the direct challenge to information processing professionals of increasing data processing service and quality while minimizing cost. The relevance of this theme is reflected in the topics of recent news accounts and magazine articles touching on the management challenge in all sectors of our economy. Congress and the President clearly set forth the challenge as a goal for the Federal Government in the Paperwork Reduction Act of 1980:

> "automatic data processing and telecommunications technology shall be acquired and used in a manner which improves service delivery and program management, increases productivity, reduces waste and fraud, and reduces the information processing burden on the public and private sectors."

Productivity growth in both the private and public sectors is an important factor in achieving our stated national goals of:

— reducing inflationary pressures,
-- raising living standards,
-- making U.S. goods competitive in world markets, and
— protecting the quality of the environment.

Productivity is defined as the relationship between the quantity of goods or services produced (output) at a given level of quality and the resources used (input). This relationship can be measured over time for an organizational unit or for an economy as a whole by comparing a specific period with a preselected base period. Productivity improvement, therefore, is an increase in the ratio of outputs to inputs; that is, a higher quantity and/or quality of goods or services at the same cost, or the same quantity of goods or services at a lower cost, without sacrificing quality.

It is obvious that productivity improvement is important. For a company, productivity means producing at a lower cost than the competition. For a government, productivity means providing more service for a tax dollar. For a nation, productivity means improving the standard of living, controlling inflation, and competing effectively in world markets. From 1945 to 1970, U.S. productivity increased at an annual rate of 3.2 percent. However, from 1966 on, productivity in the U.S. grew at a much more modest rate of 1.6 percent, and in 1978, the rate slipped below 1 percent. This trend continued in 1979, when productivity actually declined by almost 1 percent.

The major factors which affect productivity growth are:

Human resources; constituting 70 percent of the input cost in the U.S.economy. Business Week projects that 45 million jobs will be affected by increasing automation before the year 2000 -- that's half of the present total.

Technological progress; better products, processes, and systems. Promises of office automation and workflow steamlining challenge management to improve the productivity of the 55 million white collar workers in the U.S. (white collar worker productivity increased only 4 percent from 1968-1978, while blue collar productivity improved 84 percent during that period).

Investment and economic growth; new tools and capital equipment and expanding or declining market for outputs. Business Week recently projected a tripling of U.S. investment in computer-aided design and computer-aided manufacturing equipment to about $5 billion in 1985. The greatest impediment to reaping the productivity gains from these tools was seen by many experts to be software development. The theme of CPEUG 80 -- CPE Trends in the 80's -- reflected on the spectacular growth, progress, and changes in computers and computer applications in the 70's as well as future trends of the 80's. This year's theme suggests where the direction of computer usage and performance improvement efforts should be going. Increasing organizational productivity will be a focus of both Federal and commercial management for years to come. Our contributions to both increased efficiency and effectiveness in computer utilization -- from microcomputers to supercomputers to digital switches in communications -- will be of much importance in the drive for increased productivity. Our sensitivity and ingenuity in increasing the productivity of the user and his results, not just high CPU utilization levels and greater channel utilization, will be a key factor in our organizations' ability to succeed in increasing productivity.

I hope you both enjoy and greatly benefit from this year's conference. A truly fine team on this year's conference committee dedicated their time, talent, and effort unstintingly to bring it about. To all of them, their names appear elsewhere in these proceedings, my sincerest thanks for a job well done. I especially thank Peter J. Calomeris for an excellent job of keeping us to our deadlines and the painstaking job of arranging for all of the publications. A special thank you also goes to our hosts -- the U. S. Air Force, Air Training Command.

Carl R. Palmer
CPEUG 81 Conference Chairperson
November 1981

CPEUG 81

Preface

The theme of CPEUG 81 directly addresses the principal challenge of the 80's -- INCREASING ORGANIZATIONAL PRODUCTIVITY. To meet this challenge, information managers and performance specialists must ensure that their organizations increase the quantity and quality of automated information services while minimizing costs. The Conference program reflects the critical role of information services in the productivity and survival of today's organizations, as well as such trends as increasing personnel costs, limited budgets, and the convergence of data processing, word processing, and communications technologies. The keynote address (Business Automation and Productivity -- Fact, Fiction, and Prediction) and the keynote panel (Impact of Changing Technology on Productivity) will clearly focus the Conference on this challenge.

As in previous Conferences, CPEUG 81 has expanded the size and scope of the technical program to encompass the expanding role of CPE. There are three parallel sessions instead of two, and new areas such as Office Automation are included. Track A is devoted to performance-oriented management throughout the life cycle, and includes sessions in such areas as Requirements Analysis, Capacity Planning, Systems Development, and Acquisition. Track B concentrates on special technologies (data base machines, data communications, etc.) and analysis tools/techniques important to good performance and productivity. Track B also spotlights several special application areas, including state and local government, medical, welfare, and personnel systems. Track C consists of tutorials and case studies that offer familiarization and training in the major areas addressed in the other two tracks. The program also includes three receptions and expanded Birds-of-a-Feather special interest sessions to promote the informal exchange of ideas and experiences that is a trademark of CPEUG. Many people contributed to the success of the CPEUG 81 program. Carol B. Wilson, Program Vice-Chairperson, organized the tutorial track and several sessions in other tracks. The Conference Committee, session chairpersons, authors, tutors, and referees all deserve special recognition for their time and patience. The invaluable administrative support of Sylvia M. Mabie merits special thanks.

Thomas F. Wyrick
CPEUG 81 Program Chairperson
November 1981

Abstract

These Proceedings record the papers that were presented at the Seventeenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 81) held November 16-19, 1981, in San Antonio, TX. With the theme, "Increasing Organizational Productivity," CPEUG 81 reflects the critical role of information services in the productivity and survival of today's organization, as well as such trends as increasing personnel costs, limited budgets, and the convergence of data processing, communications, and word processing technologies. The program was divided into three parallel sessions and included technical papers on previously unpublished works, case studies, tutorials, and panels. Technical papers are presented in the Proceedings in their entirety.

Key words: Benchmarking; capacity planning; chargeback systems; computer performance management; data base machines; end user productivity; human factors evaluation; information system management; office automation; performance management systems; resource measurement facility; simulation; supercomputers.

CPEUG Advisory Board

CPEUG81

Conference Committee

CONFERENCE CHAIRPERSON	Dr. Carl R. Palmer U.S. General Accounting Office (202) 275-4797
PROGRAM CHAIRPERSON	Thomas F. Wyrick Federal Computer Performance Evaluation and Simulation Center (703) 274-7910
PROGRAM VICE-CHAIRPERSON	Carol B. Wilson Fiscal Associates, Inc. (703) 370-6381
PUBLICATION CHAIRPERSON	Peter J. Calomeris National Bureau of Standards (301) 921-3861
PUBLICITY CHAIRPERSON	Theodore F. Gonter U.S. General Accounting Office (202) 275-5040
PROCEEDINGS EDITOR	Terry W. Potter Digital Equipment Corporation (617) 493-9749
ARRANGEMENTS CHAIRPERSON	Dr. Jeffrey M. Mohr Arthur Young & Company (202) 828-7176
REGISTRATION CHAIRPERSON	Alfred J. Perez U.S. Air Force Data Systems Design Center (205) 279-4051
AWARDS & VENDOR PROGRAM CHAIRPERSON	Robert G. Van Houten U.S. Army Command & Control Support Agency (202) 695-6272
LOCAL ARRANGEMENTS CHAIRPERSON	William R. Hunsicker Headquarters, ATC/ACDMX (512) 652-6471
FINANCE CHAIRPERSON	James G. Sprung The MITRE Corporation (703) 827-6446

Referees

Barbara Anderson

H. Pat Artis

Duane R. Ball

V. David Cruz

Jeffey Gee

Thomas P. Giammo

Dennis M. Gilbert

Theodore F. Gonter

Gregory Haralambopoulos

John E. Hewes

Phillip C. Howard

L. Arnold Johnson

John C. Kelly

Howard S. Kresin

David Lindsay

Bobby L. McKenzie

Jim Mulford

Paul R. Nester

Terry W. Potter

Peter Robson

Dennis Shaw

Thomas F. Wyrick

CPEUG 81

Table of Contents

CASE STUDIES

PANEL OVERVIEWS

TUTORIAL OVERVIEWS

CPEUG81

Effective Life Cycle Management
Track A

CPEUG81

Requirements and Workload Analysis

SESSION OVERVIEW

REQUIREMENTS & WORKLOAD ANALYSIS

James O. Mulford

International Computing Company
3150 Presidential Drive
Fairborn, OH 45324

Accurate definition and analysis of ADP requirements and their resultant system workload is probably the most difficult issue to be addressed by performance analysts. Requirements may be defined in numerous ways (e.g., user terminology, existing system workload, existing programs, new concepts of operations) and the performance analyst must translate these definitions into terms that can be applied in system sizing, performance analysis, benchmark definitions, or other CPE tasks. Many problems are encountered during this translation, including the following:

- large amounts of data must be analyzed
- requirements are defined in various forms
- user terminology must be understood by the analyst
- new requirements are often not fully defined

These requirements and workload analysis complexities will continue to haunt performance analysts into the 198's. In fact, expanded use of automation to improve complexities of this analysis. Fortunately, tools and techniques that aid in requirements and workload analysis are receiving increased emphasis. Advances in ADP technology (e.g., processing speed, memory size and management, telecommunications, DBMS) have allowed this renewed emphasis on new tools and techniques. Better and more complete data on system workload can now be maintained. Advanced analysis techniques (e.g., cluster analysis, data reduction and characterization) are now being applied. Automated requirements definition tools (e.g., PSL/PSA[1]) are now available.

The following papers illustrate this renewed emphasis on innovative methods for analyzing requirements and system workload. All three papers demonstrate productive, cost effective approaches which can be applied in many environments. I sincerely hope that these papers will be followed (in future years) by additional and increasingly sophisticated approaches to requirements and workload analysis.

[1] Problem Statement Language/Problem Statement Analyzer, University of Michigan

LONG-RANGE PREDICTION OF NETWORK TRAFFIC

William Alexander
Richard Brice

Los Alamos National Laboratory
Computing Division
Los Alamos, NM 87544

A method of making long-range computer system workload predictions is presented. The method quantifies the effect of qualitative changes in computing by identifying assumptions and by considering the effect of a change on individual users. The method is illustrated by an example involving message traffic in a large computer network.

Key words: Computer networks; long-range forecasting; user behavior; workload forecasting.

1. Introduction

Management planning procedures sometimes require computer system workload forecasts for five or even ten years in the future. Present workload prediction methods are inadequate at such long ranges because the changes in system use are likely to be qualitative rather than just quantitative. Perhaps a computer measurement professional should be reluctant to make such long-range predictions if, possibly, too much credence will be given them. However, when you must make these predictions, how do you proceed?

In this paper we present a method for making long-range workload predictions that quantifies the effects of qualitative changes in computing. Naturally, such predictions are somewhat speculative, and we claim only to provide a framework with which to organize and quantify assumptions. The method consists of constructing a polynomial expression for the workload in which each term represents the effects of one change. The terms are constructed by concentrating on the effect that the change will have on individual users. This method explicitly represents assumptions and allows parametric ranges of results.

Some papers on workload forecasting for management planning look at current workload analysis; others study the extent of growth or change in computing activities. Determination of the current workload is heavily represented, probably because it is the most straightforward process in forecasting. Prediction methods include measurement techniques [1-5],[1] abstraction of synthetic workloads from the measurements [4,6,7], and reduction of the measurement data to manageable magnitude, for example, clustering analysis [8-10]. A second category addresses forecasting from a management perspective. These papers attempt to determine growth or change in activities that may affect computing needs. Isolated approaches exist that attempt to bridge the gulf between qualitative changes in the activities and their quantitative effects on computer resource requirements. Prediction methods in this area include extrapolation from resource requirements of existing application programs [11-13], forecasting of resource requirements for applications that are not yet completely implemented [14,15], and also some effects of

[1] Figures in brackets indicate the literature references at the end of this paper.

feedback between workload and level of service provided to users [16].

The essence of these approaches is to determine the nature of the current computing workload and, using this information, to project the amount of similar work that will be done at some future time. There are differences in how the current workload determinations are made and in the fundamental units of measure used to describe the workload. The units of measure range from resource utilization data for specific computer components to characterizations of project activities. These approaches seem best suited for short term (1-2 year) forecasts, because the effects of quantitative changes are likely to outweigh the effects of qualitative changes during this interval.

These methods are not suited to our specific problem, which is to forecast effects of qualitative changes in computing. In particular, these approaches do not address the influence that revolutions in computing hardware and services exert on how a user does his work. A second difficulty in forecasting is that long-term forecasts are almost certain to be wrong. This difficulty suggests that these forecasts should be cast in a form that is easy to update as new information arrives. Some problems that may occur if updating is not anticipated are described in Reference 17.

In Part II, we describe the need to predict message traffic in the Los Alamos National Laboratory (Los Alamos) Integrated Computing Network up to 1990. In Part III, we explain our method and illustrate its use.

2. The Problem

2.1 Integrated Computing Network

At Los Alamos, the Central Computing Facility (CCF) includes an Integrated Computing Network (ICN) that allows all validated computer users at the Laboratory access to almost any of the machines or services of the CCF.

Figure 1 is a schematic diagram of the ICN. At the "front end" of the Network (the right side of the diagram) an arbitrary

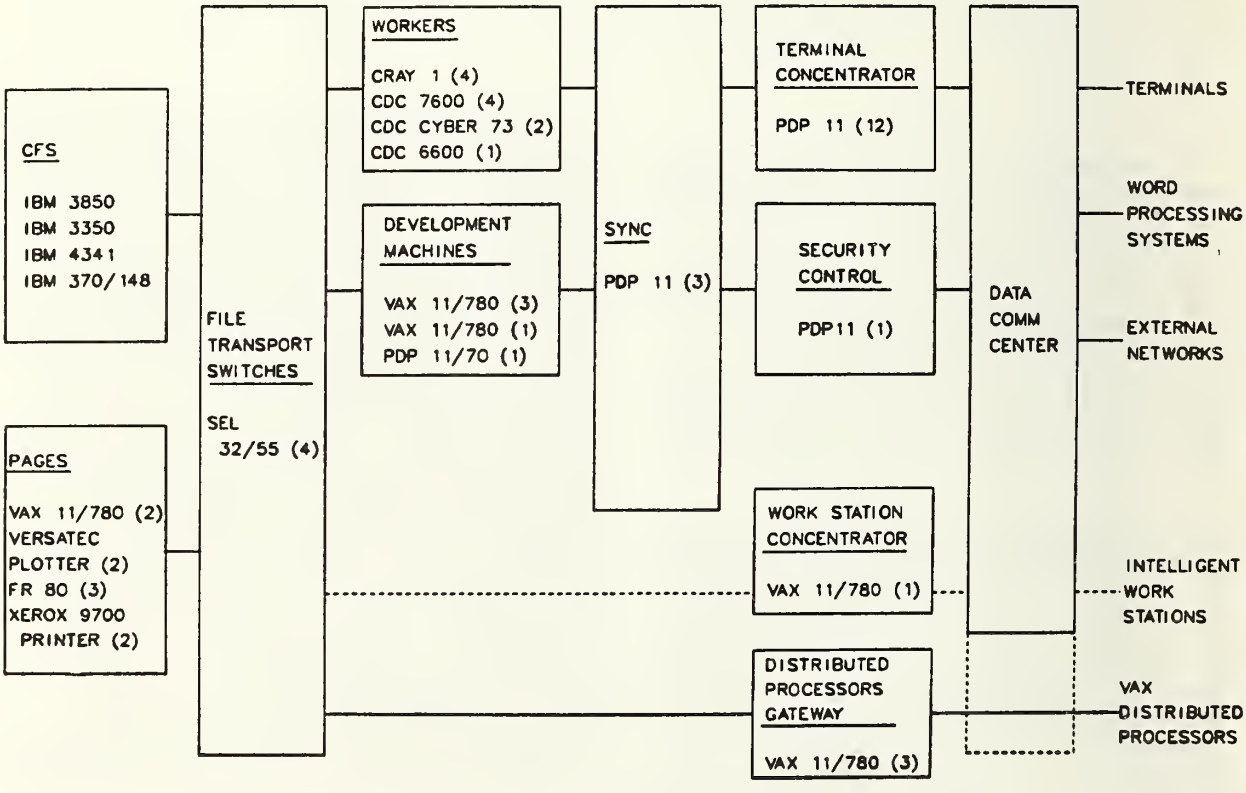


Figure 1. Functional Diagram of the ICN

number of terminals (currently about 1350) and remote entry stations are concentrated in stages to front end switches (the SYNCs), so that traffic can be routed between any terminal and any worker computer. Thus, aside from administrative restrictions, a user can log in on any worker from any terminal. The worker computers include three Cray-1s, four CDC 7600s, one CDC 6600, and two CDC Cyber-73 computers. Each of the worker computers is connected to the File Transport (FT) switches and, by the FT switches, to the "back end" of the Network (left side of the diagram). The FTs allow the workers to send files to each other and to the special service nodes in the Network. The special services provided by the Network at present include

- o an output station (PAGES) to which are attached a wide variety of printing and graphics devices,
- o a mass storage and archival facility (CFS) [18], and
- o XNET, which handles file traffic between workers and computers outside the ICN.

Messages between workers and SYNCs are usually quite small and are never larger than 1000 bytes. Messages routed through the FTs can be as large as 25,976 bytes; large files are broken by the sending machine into messages no larger than this, and the messages are sent sequentially (the ICN is not a packet-switching network).

In this paper, a "message" is one user- or program-defined group of bytes (plus network header) transferred together from a source node to a destination node in the ICN. "Nodes" include terminals, worker computers, and special service stations, but not concentrators or switches (SYNCs and FTs). From the point of view of network implementation, messages are certainly the appropriate unit of workload. From a larger view, considering the ICN as a unit, one might first think of workload in terms of terminal sessions, tasks submitted for execution on worker computers, etc. We believe that messages are an additional valid measure of workload, because there is a fairly direct correspondence between user or user program commands and messages generated. Messages result from a carriage return at the terminal and from certain explicit program functions or worker computer command language commands.

With colleagues we have just begun a new network performance measurement and evaluation project on the ICN. This project includes measurement and characterization of message traffic in the Network and analytic and simulation models of the Network. With these models we are beginning to identify the critical resources in the Network as well as to investigate the effects of increased traffic load, new equipment, and alternate configurations. Both the measurements and the models are at present rather crude.

In some of the following analysis we treat short messages (less than 100 bytes) and long messages separately, because our models indicate that different resources are critical in handling them. The critical resource limiting the Network's capacity to carry large messages seems to be buffer space in the switches, while line capacity and switch processor capacity are critical for small messages.

At present there are about 3000 users of the CCF. We measure approximately 20 large and 80 small messages per second in the back end of the Network and about 100 small messages per second in the front end. This is 0.06 small and 0.007 large messages per second per user.

2.2 The Forecasting Assignment

Recently we were asked by management to predict what the network traffic in the ICN would be at various points in the future up to the year 1990, 10 years from now. Current management forecasts indicate that the number of users of the Network will grow linearly from the present 3000 to 5700 in 1990; managers also anticipate a certain number of large worker computers in the Network by that year.

If the kind of work people do and how they go about doing it both remained constant, then the problem would be relatively straightforward. We might, for example, simply predict that the load in 1990 would be

(5700/3000) * (present load)

ignoring the different number and kinds of worker machines in 1990 on the assumption that messages are generated by programs and people, not primarily by machines. However, computing habits have changed significantly in the past 10 years, and they are likely to again in the next 10. Timesharing radically altered the way people used computers in the 1970s, distributed processing and networks are doing it now, and there may be time for two more revolutions by 1990. Change seems to be a given in computing, and no one has developed a model to predict it. Thus we preceded our response with numerous caveats, and, when management promised to heed them, [illegible]t to work.

Clearly, the traditional PME predictive tool, namely a model of the Network, does not apply directly to this problem; models are designed to take workload as input, not to predict it. Furthermore, there exists at present no characterization of our computer workload in terms of "worksteps" or "activity units" [12,13], nor any formula for translating from these to network activity. Finally, even if we had such workload characterization and such a translation formula, it is not clear that the formula would be valid for computing conditions 10 years hence. In fact, the nature of the problem and the lack of data force us into the role of futurists, a role for which a systems analyst may be no better qualified than the next person.

3. The Solution

3.1 The Method

The central idea of our method is to concentrate on the individual user, that is, to predict the effect on the user of future changes in network equipment, topology, and services. This is clearly risky, because people are the least understood and least predictable element in computing systems. Nevertheless, this focus seems necessary, because we do, in fact, believe that network traffic is affected more by what people choose to do and how they choose to do it than by the equipment they use. Of course, network topology, equipment, and services make certain tasks easy and others more difficult, but so do other factors. We are not trying to literally predict human behavior; we are trying to orient and focus our thinking in the face of too much uncertainty.

The first step is to identify factors that will change computing in our network. Then we quantify the effect of each factor on network traffic that individual users generate. Finally, we collect the terms representing each factor into a polynomial expression.

3.2 Five Factors

We were able to identify five factors that we believe will affect the way people use the ICN in the next few years. They are as follows.

1. Specialization of the Network. At present, CFS and PAGES are specialized nodes to which users from any worker can send files for permanent storage or for output. In the future, specialized nodes for word processing, for a network status and performance data base, and for other unanticipated functions may exist. (In fact, word processing software is available on a PDP-11/70 in the Network now, but this software is not yet widely used.) In addition, the worker computers themselves may become more specialized with some machines serving mostly as number crunchers and others as general-purpose front ends to the number crunchers.

2. Increased use of intelligent and graphics terminals.

3. Proliferation of distributed processors (DPs) and local networks of DPs within the Laboratory but outside the ICN. For a variety of reasons, the number of mini- and midicomputers outside the ICN continues to grow. They are used both for specialized purposes, such as process control, and for general computing; some are connected in small local networks. Typically these can communicate with any node in the ICN via XNET.

4. Electronic mail. Some electronic mail system will probably be installed at the Laboratory within the next few years, although it may be implemented as a separate mechanism rather than through the ICN.

5. Connections with remote networks. The most likely candidates are the computing facilities at other Department of Energy laboratories. Since these installations tend, at present, to have sufficient computing power for their own needs, the connections will probably be used to transfer data, programs, reports, etc., rather than to allow remote use of our computers. Similar connections to additional networks are possible.

Each of these five factors is either a trend that we see now in computing at Los Alamos or a capability currently being discussed and considered for inclusion here. In other words, we did not attempt any serious long-range crystal ball gazing, although the method allows this if you have the courage (see Section E). In the next section, we discuss the effect of each of these five factors on network message rates.

3.3 Analysis of Factors

It seems easiest to break the estimation of the effect that a change will have on any

system measure into two steps. First, one can analyze the qualitative aspects of the effect. For example, is the effect most naturally expressed as a ratio to the present number of messages a user generates or as an addition to that number? Is it independent of the user's current activity? Is it independent of the number of users? Answers to these questions will determine the position of the factor, which represents a given change in the polynomial formula for computing the value that the measure is expected to have in the future. The second step is then to plug in a numeric value for each factor, or perhaps a range of numeric values.

We will illustrate this two-step process for each of the factors described in the previous section.

1. The specialization of the Network will clearly increase message rates. As specialized service nodes are added one by one, an individual user doing tasks functionally equivalent to present tasks will generate, perhaps even unknowingly, more network messages as his files are shipped to these nodes. The portion of a user's messages due to specialization will grow in proportion to the increased specialization of the Network. Therefore, a formula for the number of small messages in the Network should contain a multiplicative factor a in a term

   a*m*Ny,

   where Ny is the number of ICN users in the future year in question and m is the observed rate of small messages per user today. That is, specialization will increase small messages per user per unit time by some factor a. There will be a similar term

   A*M*Ny

   in the formula for large messages. The way specialized nodes are now used indicates that the users will mostly ship large files that will appear as large messages; this is partly a matter of economics. For every large message in our network there is at least one small protocol message, so that the absolute increase in the two types may be about equal; however, because there are presently more small than large messages, A is greater than a.

   If we observe that 80% of large messages currently go to or from specialized nodes, and if we believe that a user will generate 50% more messages because of network specialization by year y, then the value for A in the formula for that year should be 1.4. We might plug in values of 1.2, 1.4, and 1.8 to get a range of answers corresponding to a range of assumptions about future network specialization.

2. The effect of intelligent and graphics terminals will be limited almost entirely to the front end of the Network. The use of graphics terminals will increase the large message rate from workers to terminals, because terminal output will sometimes consist of plot information for a full screen instead of one line of text. The effect of intelligent terminals, whether graphics or not, may be complicated. On the one hand, the ability to do local processing, especially screen editing, should result in fewer messages of much larger average size. On the other hand, some users may program their terminals to issue very frequent program or network status checks on a background basis and take some action only when a certain response is obtained, thus greatly increasing the small message rate.

   In any case, the factors b and B representing this effect should probably be multiplicative as are a and A above. Management projections indicate that 1000 of the terminals in the Lab will be intelligent in 10 years. We have observed that, at present, about one-fourth of all terminals are logged in on any morning. An assumption of an upper bound of 2.5 large messages per minute at these terminals gives 625 large messages per minute, which is about half the present rate; thus, we used values of from 1.1 to 1.5 for B. We used values of from 0.9 to 1.1 for b. The small range of values for b indicates that not all terminals will be intelligent and that most messages are already small.

3. The increased use of distributed processors and of local networks will certainly decrease the ICN message rate per user. Almost all of these users' terminal traffic, which consists mostly of small messages, will be eliminated from the ICN. They will still use the ICN for executing large

programs prepared locally and for special services mostly involving large files.

Once again, we decided that the factors c and C should be ratios of the present message rates per user. Values of from 0.5 to 1 for C and 0.25 to 1 for c seem reasonable.

4. If electronic mail is implemented using the ICN, then, obviously, message traffic will increase. It is not at all clear that there is any correlation between the rate at which users currently generate messages and the rate at which they will receive mail. However, mail traffic will probably be proportional to the number of people using the system. We assumed the relationship will be linear (although there are certainly other plausible possibilities). Thus we included terms

   d*Ny and D*Ny

   in the formulas for the number of small and large messages. We eventually decided that people would send and receive less than five large mailings per day, which is a negligible addition to our load; therefore, we used the value zero for D.

5. The additional message traffic caused by connecting our network to others would depend very much on the administrative nature of the connection. If remote users were given essentially the same capabilities as local users, then the appropriate adjustment to the formulas is simply to increase the value of N by the number of remote users. If use of the connection is restricted to sharing programs, data, and reports between sites, in other words, if the link is used as a fast substitute for the Postal Service, then the message rate might be independent of the number of users altogether and might depend instead on programmatic schedules. We assumed that the latter was more likely and added a simple term e to each formula to account for some small constant number of messages due to this connection.

3.4 Formulas

Collecting all the terms defined in the previous section resulted in the following formulas:

$$SMy = a*b*c*m*Ny + d*Ny + e \quad (1)$$

$$LMy = A*B*C*M*Ny + D*Ny + E \quad (2)$$

where

SMy and LMy	are the number of small and large messages per second in the ICN in year y;
m and M	are the current (1980) number of small and large messages per second per user;
Ny	is the number of CCF users that year;
a	is the factor by which network specialization will affect the number of small messages per second per user that year;
b	represents the effect of intelligent terminals;
c	represents the effect of distributed processing;
d	is the number of small messages per user per second due to electronic mail;
e	is the number of additional small messages per second due to connections to external networks; and
A, B, C, D, and E	are the corresponding factors for large messages.

We can now plug various values for each of the factors into the formula and get "best guess," "worst case," and other values for message traffic. We can also experiment with the effects of particular assumptions; for example, we can assume that all terminals will be intelligent in 10 years or that electronic mail traffic is proportional to the square of the number of users. We can investigate "disaster" scenarios; to illustrate, we can determine the rate at which intelligent terminal owners would have to generate status queries to the Network to saturate its message handling capacity. Finally, we can determine by inspection or by trial which assumptions are most critical, for example, the above formulas are clearly more sensitive to the value of a than to the value of e.

3.5 Other Possible Factors

The five change factors discussed above are certainly not the only ones that will affect computing in the Laboratory in the next few years. Since we constructed the above formulas, we have learned that, unknown to us, others in the Laboratory were already planning another change, namely a Laboratory-wide automated information management system (AIMS). Some of the pieces of such a system, such as accounting programs and some inventory programs, are already run on worker computers in the ICN. Their integration into a comprehensive, widely used management information system would certainly increase network message rates. The point of this example is that as many people as possible, from a variety of disciplines, should be included in the process of thinking of changes in computing.

More speculative changes than those we have given might also be included in a projection. Very powerful processors on a single chip will soon be available at very low cost. The use of high-quality graphics output devices may become much more widespread at the Laboratory to display movies (16 frames of graphics output per second) used to study simulation modeling programs. Although present worker computers are not capable of producing 16 frames per second from these programs, long sequences of frames could be generated and stored in CFS; these could be fetched and fed to the graphics device by the cheap powerful processor at such a rate. If this happens, it will greatly increase the large message rate.

4. Conclusions

Inserting our "best guess" factor values into the above formulas resulted in message rates for 1990 of five to six times the present observed rates. To anyone familiar with the history of computing, it might seem unlikely that any workload measure on any system will grow by "only" 500% in 10 years. If this projection, in fact, turns out to be low, the reason will probably be that we failed to anticipate some development in computing that radically affects network use. The necessity of anticipating such changes is, of course, the greatest weakness of our method; however, this weakness is inherent to the problem. It can be overcome somewhat by requesting input from as many people as possible.

Our method of prediction presented in this paper identifies specific assumptions. It allows experimenting with different values of factors to see the part each plays in the total prediction. More accurate data about the effect of a given change can be easily incorporated into the formulas so that predictions grow more accurate in an evolutionary way. Concentrating on the effects on individual users might also work well for short-term predictions, but we found this method especially helpful as a way of isolating and organizing the uncertainties and shakey assumptions inherent in long-range prediction.

References

[1] Lindsay, D. S., A Hardware Monitor Study of a CDC KRONOS System, Proceedings, International Symposium on Computer Performance Modeling Measurement and Evaluation, Harvard University, March 1976, pp. 136-144.

[2] Partridge, D. R. and Card, R. E., Hardware Monitoring of Real-Time Computer Systems, Proceedings, International Symposium on Computer Performance Modeling, Measurement and Evaluation, Harvard University, March 1976, pp. 85-101.

[3] McDougall, M. H., The Event Analysis System, Proceedings, 1977 SIGMETRICS/CMG VIII, Washington, D.C., November 1977, pp. 91-100.

[4] Keller, T. W. and Lindsay, D. S., Development of A Performance Evaluation Methodology for the CRAY-1, Proceedings, 1977 SIGMETRICS/CMG VIII, Washington, D.C., November 1977, pp. 169-176.

[5] Abrams, M. D. and Neiman, D. C., NBS Network Measurement Methodology Applied to Synchronous Communications, Proceedings, CPEUG80, Orlando, Florida, November 1980, pp. 63-70.

[6] Hughes, J. H., A Functional Instruction Mix and Some Related Topics, Proceedings, International Symposium on Computer Performance Modeling Measurement and Evaluation, Harvard University, March 1976, pp. 145-153.

[7] Terman, F. W., A Study of Interleaved Memory Systems by Trace Driven Simulation, Proceedings, Symposium on Simulation of Computer Systems, Boulder, Colorado, August 1976, pp. 3-10.

[8] Mamrak, S. A. and Amer, P. D., A Feature Selection Tool for Workload Characterization, Proceedings, 1977 SIGMETRICS/CMG VIII, Washington, D.C., November 1977, pp. 113-120.

[9] Agrawala, A. K. and Mohr, J. M., The Relationship Between the Pattern Recognition Problem and the Workload Characterization Problem, Proceedings, 1977 SIGMETRIC/CMG VIII, Washington, D.C., November 1977, pp. 131-139.

[10] Artis, H. P., A Technique for Establishing Resource Limited Job Class Structure, Proceedings, CMG X, Dallas, Texas, December 1979, pp. 249-253.

[11] Linde, S. and Morgan, L., Workload Forecasting for the Shuttle Mission Simulator Computer Complex at Johnson Space Center, Proceedings, CMG XI, Boston, Massachusetts, December 1980, pp. 10-15.

[12] Mohr, J. and Penansky, S. G., A Framework for the Projection of ADP Workloads, Proceedings, CMG XI, Boston, Massachusetts, December 1980, pp. 5-9.

[13] Perlman, W. and Jozwik, R., DP Workload Forecasting, Proceedings, CMG X, Dallas, Texas, December 1979, pp. 469-475.

[14] Smith, C. and Browne, J. C., Modeling Software Systems for Performance Predictions, Proceedings, CMG X, Dallas, Texas, December 1979, pp. 321-342.

[15] Smith, C. and Browne, J. C., Performance Specifications and Analysis of Software Designs, Proceedings, Conference on Simulation Measurement and Modeling of Computer Systems, Boulder, Colorado, August 1979, pp. 173-182.

[16] Axelrod, C. W., The Dynamics of Computer Capacity Planning, CMG Transactions, Number 29, September 1980, pp. 3.3-3.21.

[17] Tomberlin, R. D., Forecasting Computer Processing Requirements: A Case Study, Proceedings, CPEUG80, Orlando, Florida, November 1980, pp. 255-261.

[18] Blood, M., Christman, R., and Collins, B., Experience with the LASL Common File System, Digest of Papers from Fourth IEEE Symposium on Mass Storage Systems, Denver, Colorado, April 1980, pp. 51-54.

FUNCTIONAL GROUPING OF APPLICATION PROGRAMS IN A TIMESHARING ENVIRONMENT

Paul Chandler, CCP, CDP

Wilson Hill Associates
1025 Vermont Ave., N.W., #900
Washington, DC 20005

The ability to adequately describe the computing environment is important in the selection of timesharing services. A method is presented by which the user can functionally describe the workload in terms of requirements for processing, data input, and data output. Large numbers of programs can thereby be reduced to only eight functional categories which are sufficient to characterize the workload. Unsophisticated non-ADP oriented users who are unable to use traditional performance or resource oriented groupings can use this functional method satisfactorily.

1. Introduction

Procurement of a timesharing service can be a lengthy and costly process in many large procurement efforts. The primary objective of such procurements usually is to provide the most cost effective computer system which will meet both current and future needs.

The acquisition of a computer system presupposes that the workload is well defined. Typical parameters used include turnaround time, core size or segment, CPU time, number of I/O transfers, number of lines printed, etc. However, many of these parameters are available only through specially implemented hardware or software monitors. These monitors require knowledgeable ADP personnel sensitive to procurement needs, as well as a thorough understanding of system requirements.

Many procurement efforts suffer from a lack of specific resource or performance areas and are left with only a functionally-oriented benchmark which does not lend itself well to a lowest cost acquisition. Also, many procurements must deal with a sizable workload and cannot simplify their workload easily. Additionally, the procurement specialist who is conducting the computer acquisition process may be far removed from daily processing requirements or else be faced with relatively unknowledgeable ADP users who do not understand the details of machine processing.

This inability to characterize the workload is the subject of this paper. Outlined is a method by which the benchmark workload can be segmented by a reduction in the number of programs into distinct classes.

2. Methodology

This method provides for a functional grouping of application programs using three representative criteria. A dichotomous classification exists already in the minds of many ADP personnel of I/O versus CPU. This method separates the "I" from

the "O" and uses the three parameters of

- o Processing
- o Input
- o Output

By using a selection method by which each criteria is judged to be either an important or less important functional attribute of each application program, a clustering of programs emerges which share similar functionally oriented characteristics.

Figure 1 shows the eight possible ratings using the three criteria described.

Class ID	Rating Factors CPU	Input	Output
A1	Low	Low	Low
A2	Low	Low	High
A3	Low	High	Low
A4	Low	High	High
A5	High	Low	Low
A6	High	Low	High
A7	High	High	Low
A8	High	High	High

Figure 1. Criteria Classes

The assignment of application programs to a class depends upon the relative rating of each of the three criteria to the application. In order to perform this classification, standardized and descriptive data must be collected for each application system.

Data typically can be collected by means of a comprehensive data collection effort which results in each application being described in functional terms. Such a description will include a narrative description of the program's primary purpose, an estimate of required data input and output, and general qualitative descriptions of program processing.

Additional data may be collected for some quantitative data such as number of files, number of records or transactions processed, size of the data base, language type, and number of tape mounts.

Not included are specific performance oriented data such as speed of execution or elapsed seconds for a file I/O, or resource oriented data such as CPU seconds, core size used, number of bytes transferred from disk to core, etc. Only generally available data obtained from a knowledgeable user is required without specific hardware or software monitoring.

Following this qualitative data collection, each of several knowledgeable ADP persons reviews the collected data. These persons need not be users or persons from whom data was collected. These persons then make a binary high/low choice for each of the three criteria identified earlier for each application program. Each person is asked to work alone for the initial selection. Upon completion, for each person, there will be a rating for each program as to their evaluation of program functionality in each of three categories.

The ratings of each application program are compared and the persons asked to reevaluate their assignments if conflicts arise. Additionally, some information as to data or other interdependencies between programs might be used to classify an application, as well.

The result of the selection process is that for each application there are three descriptors. For the binary high/low judgment required for this model across all three criteria, eight different combinations are possible. Figure 2 shows these eight combinations in a spatial relationship.

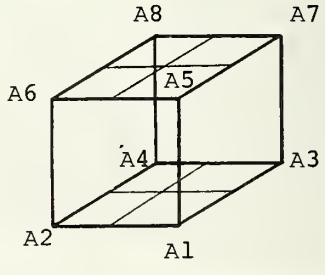


Figure 2. Spatial Relationship of Criteria

Each of these dimensions can be thought of in a spatial relationship in such a way that the selection of these criteria will define a larger group of applications.

3. Application of Method

This method anticipates that a natural grouping of programs will occur, and that groups of applications will tend to have similar characteristics based solely on the qualitative ranking of individual applications.

An example is drawn from a recent Federal procurement effort of around 6.5 million dollars including 96 application programs in a commercial timesharing environment. This procurement effort was characterized by a wide geographic dispersion of the user community, little or no information about application systems, or any other good, well-defined quantitative measurements.

Following the application of this method, the 96 programs were classed in eight functional categories out of which one or two programs could be selected for benchmarking. Figure 3 shows the distribution of these programs across groups.

ID	Group Name	Number of Applications	Rating Factors CPU	Input	Output
A1	System Support	19	Low	Low	Low
A2	Display Oriented Output	17	Low	Low	High
A3	Data Entry	1	Low	High	Low
A4	Information Manipulation	7	Low	High	High
A5	Basic Analysis	29	High	Low	Low
A6	Display Oriented Analysis	9	High	Low	High
A7	Complex Data Entry/Edit	7	High	High	Low
A8	Complex Data Manipulation	7	High	High	High

Figure 3. Group Distribution

Also included in this example is an assignment of a group name for each of the eight classes. The paragraphs describe each functional group as used in this illustration. Similar names and characteristics could be defined for another set of applications.

A1 - System Support

This group comprises those application programs with little or no requirement for either CPU, input or output. These application systems tend to be either in use at remote sites or of an infrequent nature, or used at a central location for selected special purposes. Included are those applications which query current operating system status, perform rudimentary reports or text storage of small files, or simple computational programs.

A2 - Display Oriented Output

This group relies to a greater extent on some kind of formatted output either in tabular, report, or plotted form. Computational usage appears to be low, and most applications seem to be parameter driven with small operator input required. The data bases used for this group appear to be fixed and well defined with infrequent updates. Also included are applications with high volume, repetitive type output.

A3 - Data Entry

Although only one program was classed in this group for the applications surveyed, this group might be characterized by those programs which are primarily input dependent, and which include a substantial processing of data in file building.

A4 - Information Manipulation

This group depends on a fair amount of operator supplied input and produces voluminous output. Computational resources are used in the building of file indexes, but no inter-file relationships or permanent index structure (such as in DBMS applications) occurs. Simple statistical methods may be used in some applications.

A5 - Basic Analysis

This group contains those applications in which a fairly sophisticated computational usage of machine resources is needed, either through direct number crunching or in the building of file indexes. Both input and output are equally weighted, but form a low part of the resources used.

A6 - Display Oriented Analysis

This group represents a sophisticated usage of output facilities to the exclusion of operator supplied input. Proprietary display packages are used in some applications. The computational resources used appear to be high.

A7 - Complex Data Entry/Edit

Unlike the Data Entry group (A3), these applications utilize on-line input of data with on-line correction and validation of input. The computational usage is also high. Output is a secondary consideration here from the user viewpoint, since the primary purpose of these applications is to facilitate data entry and validation.

A8 - Complex Data Manipulation

This group was judged to have the highest need and use of all three criteria measured. The overriding concern addressed here is the apparent need for high volume input and output together with complex data manipulation or index structures.

Once the initial grouping was made, a second pass was made of each group in order to generalize common features among programs. Of the 96 applications to which this method was applied, less than 7 of these had to be reevaluated, and then only a shift in one criterion resulted.

4. Discussion

A possible problem with the validity of this method might occur when only small application program environments are considered. In the example presented, a good distribution occurred across all groups but one, and the distribution appeared to be non-random. In procurement efforts where there are only a limited number of programs, it might be difficult to be certain that the best grouping had been achieved.

The familiarity of the evaluation with ADP was initially considered to be a potential problem, but did not prove to be insurmountable in this study as knowledgeable personnel who had some ADP background were available. In those instances where the evaluators are non-ADP oriented, it seems likely that additional qualitative data must be collected.

An interesting extension of this method would include the quantification of each workload criteria into small discrete steps. Each step could represent an increment of the respective criteria such that all values between an arbitrarily low and high point would be represented. The workload could then be divided up using the method presented here.

For the example given, by far the greatest time was spent in data collection. The actual evaluation of applications only required a few days. It seems likely that those procurement specialists faced with knowledgeable ADP persons will be in a better position than those faced with unfamiliar users.

In the environment where there is a scarcity of information, this

approach appears to offer some help in workload quantification. This method provides a method acceptable from a procurement specialty where a savings of time and money can be obtained in environment where there is little information.

The author is grateful to Dr. Shahla Butler for her inspiration in this effort, and to Thomas Mink for his review of the manuscript.

References

[1] Bayraktar, A. N. Computer Selection and Evaluation, June 1978, U.S. Dept. of Commerce, NTIS #AD-A061-070.

[2] Ferrari, Domenico, Computer Systems Performance Evaluation, Prentice-Hall, 1978.

[3] National Bureau of Standards, Guidelines for Benchmarking ADP Systems in the Competitive Procurement Environment, FIPS PUB 42-1, May 15, 1977.

[4] National Bureau of Standards, Guideline on Constructing Benchmarks for ADP System Acquisitions, May 19, 1980.

CLUSTER ANALYSIS IN WORKLOAD CHARACTERISATION FOR VAX/VMS

Alex S. Wight *,**

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

*Permanent address:Department of Computer Science ,University of Edinburgh,Edinburgh, U.K. EH9 3JZ

**This work was done while the author was with Digital Equipment Corporation,Maynard, MA 01754

Accurate models of computer system workloads are desirable for system modelling, performance evaluation studies, capacity planning, benchmarking, tuning and system selection. The use of cluster analysis for workload characterisation has received much attention recently. In this paper we explore the use of cluster analysis for characterisation of workloads on VAX systems running the VMS operating system. In a largely interactive environment we take as our basic workload unit the program or executable image invoked by a command from a terminal user e.g. file directory manipulation, editor, compiler, application program. We discuss some of the problems of cluster analysis and the possible uses of cluster descriptions.

Key words: Cluster analysis, Workload characterisation.

1. Introduction

There are many reports of studies of computer system workload characterisation [3,4,5,6,10,13,24,25,27,31]. Accurate models of workloads are desirable for system modelling, performance evaluation studies,capacity planning, benchmarking, tuning and selection. The use of cluster analysis in this work is not new [28]. Clustering algorithms have been developed as tools in a wide range of disciplines where extensive data analysis and recognition and classification of patterns are important. There exist both good surveys [12,14,15,18,19,20,22] and a number of books which describe clustering algorithms and the analysis of the results of clustering [7,16,21,26,29,30,33,34,35]. The application of cluster analysis to the problem of providing concise descriptions of computer system workloads has been advanced most recently by Artis [6], Mamrak and Amer [32] and Mohr and co-workers [3,4,5].

In particular Agrawala et al [4] discuss the validity of using cluster analysis while Artis [6] and Agrawala and Mohr [3] also discuss the representativeness and usefulness of the clusters which are generated. Mamrak and Amer [32] and Hughes [27] discuss the problem of deciding which of the available, measured features can be used to describe the workload units submitted to the clustering process.

These studies illustrate the feasibility of applying cluster analysis to accounting log or system measurement data and generating concise descriptions of the workload submitted to the computer system in the measurement period. If a clear picture of a stable workload already exists we do not need sophisticated analysis to confirm it. Cluster analysis may be used either to discover if available data points lie within known or expected groupings or to discover if they do exhibit any structure or concise groupings.

There is a very large number of publications on the use of cluster analysis. Useful starting points are the papers by Dubes and Jain [18,19] which describe and examine the problems of amalgamating the results of cluster analyses, statistical analyses, intuition and insight. Kendall [30,31] makes many points advising caution in the use of cluster analysis and stressing the importance of the analyst who is familiar with the context of the data and appreciates the strengths and weaknesses of the algorithms employed. Clustering results must be interpreted with care. Diday [17] discusses some possible new approaches.

The VAX/VMS operating system running on the DEC VAX 11/780 is being used in an ever-widening range of environments. To characterise the workloads in these environments requires a comprehensive methodology. In this paper we present an initial approach to the use of cluster analysis as part of the methodology required to get useful models for a number of possibly very diverse workloads. In a largely interactive environment we take as our basic workload unit the program or executable image invoked by a command from a terminal user e.g. file directory manipulation, editor, compiler etc.

2. Tools

Our starting point was Artis's very comprehensive discussion of the use of cluster analysis for capacity planning [6]. A search of the literature [7,11,12,14,16,18,19,20,29] reveals ISODATA [8,9,23] as one of the most popular clustering algorithms. This may not be unrelated to the fact that it's heuristic nature makes analysis of it difficult! Cluster analysis which attempts to classify a population of objects on the basis of features describing each object has developed in many disciplines ranging through the biological and social sciences and engineering. With a version of ISODATA available , the example of Artis and the recurring commendation of ISODATA in the literature we chose to use ISODATA.

The standard accounting log of VMS does not provide sufficiently detailed resource usage data for characterising the submitted work at the level of individual programs. Ultimately we are interested in characterising in detail interactive users and the programs they invoke. VMS was adapted to write additional messages to the accounting log recording statistics for each executed image or command.

These statistics cover:

- CPU time
- Elapsed time
- Memory occupancy
- Disc traffic
- Terminal traffic
- Paging activity

Since it obviously imposes an additional overhead and causes interference this form of monitoring can be switched off and on for periods of interest. CPU time is not recorded accurately, but in the standard VMS form by sampling and recording with a 10 millisecond

unit. Attempting to account CPU time accurately at the 1 microsecond level raised the total data collection overhead from around 5% to over 10% and was abandoned. Records from a relevant period of an available accounting log can be extracted and used as data for statistical and cluster analysis.

As we have stated already a majority of authors in the clustering literature advocate caution in applying a chosen algorithm to new data, in particular see [15,18,19,22,30,31].

3. Testing

We proceeded as follows:

(a) Fisher's Iris data [30] has been used many times to test clustering algorithms. We checked that our ISODATA program gave results consistent with Kendall's analysis [30].

(b) We generated artificial data representing points on a 2-dimensional grid. These represented intuitive ideas of dense, diffuse and scattered clusters, clusters of odd shape and outliers. Dubes and Jain [18] give the results of applying different clustering algorithms to similar artificial data. Our results were at the same time disturbing and reassuring. It was a worthwhile exercise to gain confidence in choosing values for the ISODATA control parameters but a salutary lesson in treating all clustering results with care since it is certainly possible even with a known algorithm and "obvious" data to generate quite different clusters from runs with different parameter settings.

(c) We next proceeded to use image accounting data collected from a VAX system which was processing a known and well-defined load of interactive scripts submitted from a remote terminal emulator. This gave us the opportunity to explore the interplay of cluster analysis and performance evaluation. We had to consider the problems of feature selection and scaling and continue exploring parameter settings in this much more complex case. Here again the literature generally persuaded us to follow Artis's example and scale the data using the robust mean and standard deviation for each feature such that for each feature the data is normalised with unit variance. The BMDP statistical package was used for statistical analysis of the data. Use of robust statistics for scaling reduces the effect of outliers but one must always remember that any scaling affects the formation of clusters. For this known data the results were encouraging and the expected groupings did appear.

4. Points from the Tests

The use of a remote terminal emulator provides a desirable controlled environment for this form of testing as well as more general performance work [1,2,36,37]. In this case we can explore the effects on the clusters we see of different feature sets e.g. a resource consumption set and a rate of resource consumption set [3]. For exploratory analysis of this type one expects to iterate between clustering and examining and interpreting clusters produced. Anderberg [7] suggests generating a first set of clusters and investigating individual clusters before perhaps reclustering. Outliers can be considered at this point too. Note the difference between ISODATA which weights features equally and attempts to reduce the variance of features within each cluster formed and the approach of Agrawala and Mohr [5] where it is intended that the clusters formed reflect their "natural" shapes with different weights being given to features in different clusters as they are established. ISODATA tends to produce hyperspheres and treats all clusters identically. An "elongated" cluster may not be immediately obvious. If it is contained uniquely within a hypersphere it must be deduced by examining the variances of the features for members of the cluster. Otherwise the analyst must deduce from the collection of cluster descriptions that a series of small possibly dense clusters are in fact closely related and when viewed on a

larger scale form one identifiable group.

5. Experiment

The image accounting package was installed on a VAX system in use for software development. Over a working day 1791 image records were recorded. The following five features were chosen for the cluster analysis:

CPU time
Elapsed time
Direct I/O rate(request/CPU sec.)
Buffered I/O rate(request/CPU sec.)
Page read rate(request/CPU sec.)

Statistics for these features are displayed in Figure 1. The clustering program was run with an upper limit of 50 clusters allowed. It is clear from the statistical summaries that we are likely to reach this number. In fact after 10 iterations of trying to improve the clusters we have reduced the number of clusters to 37.

6. Cluster Summaries

At this point when we look closely at the ISODATA summary statistics we find the average distance of patterns or data points from their centroids or cluster centres to be 1.8 while the average distance between cluster centers is 31.2. The co-ordinates of a cluster centroid are the mean values for each feature of the cluster members. Note that these are distances in the transformed data space. This gives a picture of tight, well-scattered clusters. Only four of the thirty-seven have points with an average distance from the centroid greater than 3.0 compared with average distance between centroids of 31.2. For the cluster centroids the distance to the nearest centroid ranges from 1.75 to 51.2 with a mean of 8.0.

7. Validity

Dubes and Jain [19] discuss in detail the question of how valid clusters are. They feel that in general, engineering applications, including computer science, tend to ignore cluster validity and authors are happy to demonstrate that the algorithm works in some sense that they consider suitable. We agree with the observation and criticism but nevertheless proceed, taking the view that cluster analysis is an exploratory tool to be used with care in examining the available data,investigating hypotheses and seeking underlying structure. The density of our individual clusters and the way they are spread satisfies us while we examine the populous clusters in the light of our knowledge of the workload.

8. Cluster Descriptions

Figure 2 shows that the 8 clusters with the most members include 94.8% of the images. If these clusters have interpretable characteristics we have a starting point for a characterisation of the workload. The values of the centroids of these clusters are displayed in Figures 3,4,5 in terms of the untransformed values of the features. For each cluster we can look for the dominant features and for each image name we have a list of the clusters in which it appears. The groupings of the clusters can now be checked and interpreted by consulting this list. Although there are cases where images appear in many different clusters interpretation is possible. The cluster groupings of Figure 2 are roughly as follows.

[1] Clusters 1,5 and 10 are represented by average values of the features chosen and turn out to be largely calls on standard command language utilities manipulating files and directories. Such commands where they appear in other clusters are cases where they have generated a large value for a particular feature and been treated as a separate cluster since ISODATA will split a cluster if there is a large variance in one feature.

[2] Clusters 14 and 18 are dominated by demands for CPU time and high terminal traffic represented by buffered I/O. The images represented here are

editor use,mail utilities and application programs.

[3] Cluster 21 is dominated by a high page read rate The members of 21 are subsets of some of the previous images but also closely clustered.

[4] The grouping of clusters 7 and 9 is dominated by a high buffered I/O rate and these clusters contain images making heavy use of video terminals.

Rather than continue in this vein examining all the remaining very small clusters we first complete the broad picture with a more conventional statistical analysis. Returning to the raw data we list for each feature the images with the highest values. When we trace those to the clusters we find that they have been identified as clusters and often the elongation effect can be discerned i.e. a number of small clusters have the same dominant feature and are seen to be identified as nearest neighbours in the transformed hyperspace.

Turning to Figure 6 and the most frequently occurring images we find as follows. The most common image with 312 instances has all but 16 of these in clusters 1,5 and 10. Proceeding to the second and third, all except 4 of 194 are in 1,5 and 10 and all except 7 of 190 likewise. There are 77 unique images recorded and 76% of all images recorded are accounted for by instances of the 10 most frequently-occurring images.

9. Workload Modelling

The clusters found can be used as a basis for a model of the workload. For each important cluster that we identify real or synthetic programs must be available or generated with characteristics matching those of "average" cluster members. The number of occurrences of a model job in the workload model must be adjusted so that the load on the system generated by each cluster is maintained. More detailed study of the accounting log allows us to look for variations in the load over time and fluctuations in the mix of active programs. The log also contains sufficient information for commonly occurring sequences of commands to be found and incorporated in models. We do not of course get user characteristics like typing speeds and think times.

We may hope that the largest clusters formed from a series of logs allow us to model what appears to be a stable component of the workloads under study. However we must also take note of the outliers or very small clusters as they may turn out to make some of the greatest demands for system resources or be characterised by very variable activity. If the small clusters and outliers make significant demands on the system then representing their effect is important but may be more difficult if they differ dramatically from site to site, period to period and especially if they largely reflect the choice of features for the cluster generation. This form of variation is also influenced by the level of workload unit we choose and the nature of general-purpose interactive computing.

Note also that having identified clusters we must look at them in terms of features not used for clustering both in construction of models and possible amendments to the clustering and characterisation methodology.

10. Discussion

Cluster analysis has often been discussed and used for workload characterisation in general and benchmark construction in particular. Much useful work can be done with fixed workloads [1,2,36,37] but as user communities broaden and technological change speeds changes in user demands on computing systems it is desirable to attempt to generate currently representative workload models and keep them up-to-date.

This work explores part of the methodology and indicates that cluster analysis can be used for data from highly interactive systems with

a wide range of user demands as well as some of the more constrained batch environments previously reported.

Cluster analysis has developed its own literature in separate fields. There is still scope for analysis of clustering techniques for computer workload characterisation e.g. feature selection, effect of correlation between features, effects of scaling and weighting of features, algorithm comparison,validity of clusters, use of clusters, efficiency of algorithms and integration of results into a more complete characterisation methodology.

11. Acknowledgements

Many people contributed to an enjoyable and productive time at Digital. For the work reported here thanks particularly go to Colin Adams, Rick Fadden, Steve Forgey and Ted Pollak for many hours of assistance and discussion.

References

[1] Adams,J.C., Currie, W.S. and Gilmore , B.A.C., The Structure and Uses of the Edinburgh Remote Terminal Emulator, Software Practice and Experience, Volume 8, 1978.

[2] Adams, J.C., Performance Measurement and Evaluation of Time-Shared Virtual Memory Systems, Ph.D. Thesis, University of Edinburgh, 1977.

[3] Agrawala, A.K. and Mohr, J.M., Some Results on the Clustering Approach to Workload Modelling, Proceedings of CPEUG 13th meeting, October 1977. NBS Special Publication 500-18.

[4] Agrawala, A.K. and Mohr, J.M., The Relationship Between the Pattern Recognition Problem and the Workload Characterisation Problem, Proceedings of SIGMETRICS/CMG V|||, November 1977.

[5] Agrawala, A.K., Mohr, J.M. and Bryant, R., An Approach to the Workload Characterisation Problem, Computer, June 1976.

[6] Artis, H.P., Capacity Planning for MVS Computer Systems, in Performance of Computer Installions, ed. Ferrari, D. ,North-Holland Publishing Company, 1978.

[7] Anderberg, M.R., Cluster Analysis for Applications, Academic Press, 1973.

[8] Ball, G.H., Data Analysis in the Social Sciences: What About the Details?, AFIPS FJCC , Volume 27, Part 1, 1965.

[9] Ball, G. and Hall, D.J., ISODATA, A Novel Method of Data Analysis and Pattern Classification, Stanford Research Institute, 1965.

[10] Barber, E.O., Asphjell, A. and Dispen, A., Benchmark Construction, Performance Evaluation Review , Volume 4, Number 4, 1975.

[11] Bayne, C.K., Beauchamp, J.J., Begovich, C.L. and Kane, V.E., Monte Carlo Comparisons of Selected Clustering Procedures, Proceedings of Annual Meeting of American Statistical Association - Statistical Computing Section, 1978.

[12] Blashfield, R.K. and Aldenderfer, M.S., A Consumer Report on Cluster Analysis Software, The Pennsylvania State University, 1976.

[13] Boi, L., Cazin, J., Martin, R. and Bourret, P., The Use of a Synthetic Workload for the Optimal Allocation of Files on Disk Units, in Performance of Computer Installations, ed. Ferrari, D., North-Holland Publishing Company, 1978.

[14] Cole, A.J., ed., Numerical Taxonomy, Academic Press, 1969.

[15] Cormack, R.M., A Review of Classification , Journal of the Royal Statistical Society (Series A), 1971.

[16] Clifford, H.T. and Stephenson, W., An Introduction to Numerical Classification, Academic Press, 1975.

[17] Diday, E., Problems of Clustering and Recent Advances, IRIA Research Report Number 337, 1979.

[18] Dubes, R. and Jain, A.K., Clustering Techniques: The User's Dilemma, Pattern Recognition, Volume 8, 1976.

[19] Dubes, R. and Jain, A.K., Validity Studies in Clustering Methodologies, Pattern Recognition, Volume 11, 1979.

[20] Duran, B.S. and Odell, P.L., Cluster Analysis, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 1974.

[21] Everitt, B., Cluster Analysis, 2nd edition, John Wiley, 1980.

[22] Friedman, H.P. and Rubin, J., On Some Invariant Criteria for Grouping Data, Journal of the American Statistical Association, 1967.

[23] Gnanadesikan, R., Methods for Statistical Data Analysis of Multivariate Observations, John Wiley and Sons, 1977.

[24] Hall, D.J. and Dev Khanna, The ISODATA Method Computation for the Relative Perception of Similarities and Differences in Complex and Real Data, in Mathematical Methods for Digital Computers. Volume |||(Statistical Methods for Digital Computers), ed., Enslein, K., Ralston, A. and Wilf, H.S., John Wiley and Sons, 1977.

[25] Haring, G. and Posch, R., On the Use of a Synthetic Online/Batch Workload for Computer Selection, Information and Management, Volume 3, Number 3, 1980.

[26] Haring, G., Posch, R., Leonhardt, C. and Gell, G., The Use of a Synthetic Jobstream in Performance Evaluation, The Computer Journal, 1979.

[27] Hartigan, J.A., Clustering Algorithms, John Wiley and Sons, 1975.

[28] Hughes, H.D., A Study of a Procedure for Reducing the Feature Set of Workload Data, Computer Performance Evaluation in the '80s, Proceedings of CMG X|, 1980.

[29] Hunt, E., Diehr, G. and Garnatz, D., Who Are the Users? An Analysis of Computer Use in a University Computer Center, AFIPS Conference Proceedings, Volume 38, 1971.

[30] Kendall, M.G., Multivariate Analysis, Hafner Press, 1975.

[31] Kendall, M.G., The Basic Problems of Cluster Analysis, in Discriminant Analysis and Applications ed., Cacoullos, T., Academic Press, 1973.

[32] Mamrak, S.A. and Amer, P.D., A Feature Selection Tool for Workload Characterisation, Proceedings of SIGMETRICS/CMG V|||, November 1977.

[33] Pore, M.D., Moritz, T.E., Register, D.T., Yao, S.S. and Eppler, W.G., On Evaluating Clustering Procedures for Use in Classification, Proceedings of American Statistical Association, Statistical Computing Section, 1978.

[34] Sokal, R.R. and Sneath, P.H.A., Numerical Taxonomy, W.H. Freeman, 1973.

[35] Spath, H., Cluster Analysis Algorithms for Data Reduction and Classification of Objects, Ellis Horwood Ltd., 1980.

[36] Stephens, P.D., Yarwood, J.K., Rees, D.J. and Shelness, N.S., The Evolution of the Operating System EMAS 2900, Software Practice and Experience, Volume

10, Number 12, 1980.

[37] Wright, L.S. and Burnette, W.R., An Approach to Evaluating Timesharing Systems: MH-TSS A Case Study, Performance Evaluation Review, January 1976.

FIGURE 1: SYSTEM RESOURCE UTILIZATION STATISTICS

	MEAN	MEDIAN	STANDARD DEVIATION	MINIMUM	MAXIMUM
CPU (10 MILLISECONDS)	108.393	12.00	556.428	1.00	7953.00
ELAPSED TIME (SECONDS)	67.988	1.00	358.069	0.03	8784.889
DIRECT I/O RATE QIO/CPU SEC.	13.19	9.09	19.183	0.00	233.330
BUFFERED I/O RATE QIO/CPU SEC.	85.889	62.50	92.022	0.00	1364.290
PAGE READ RATE QIO/CPU SEC.	48.333	33.333	535.562	0.08	500.00

GALAXY: WORKLOAD CHARACTERIZATION

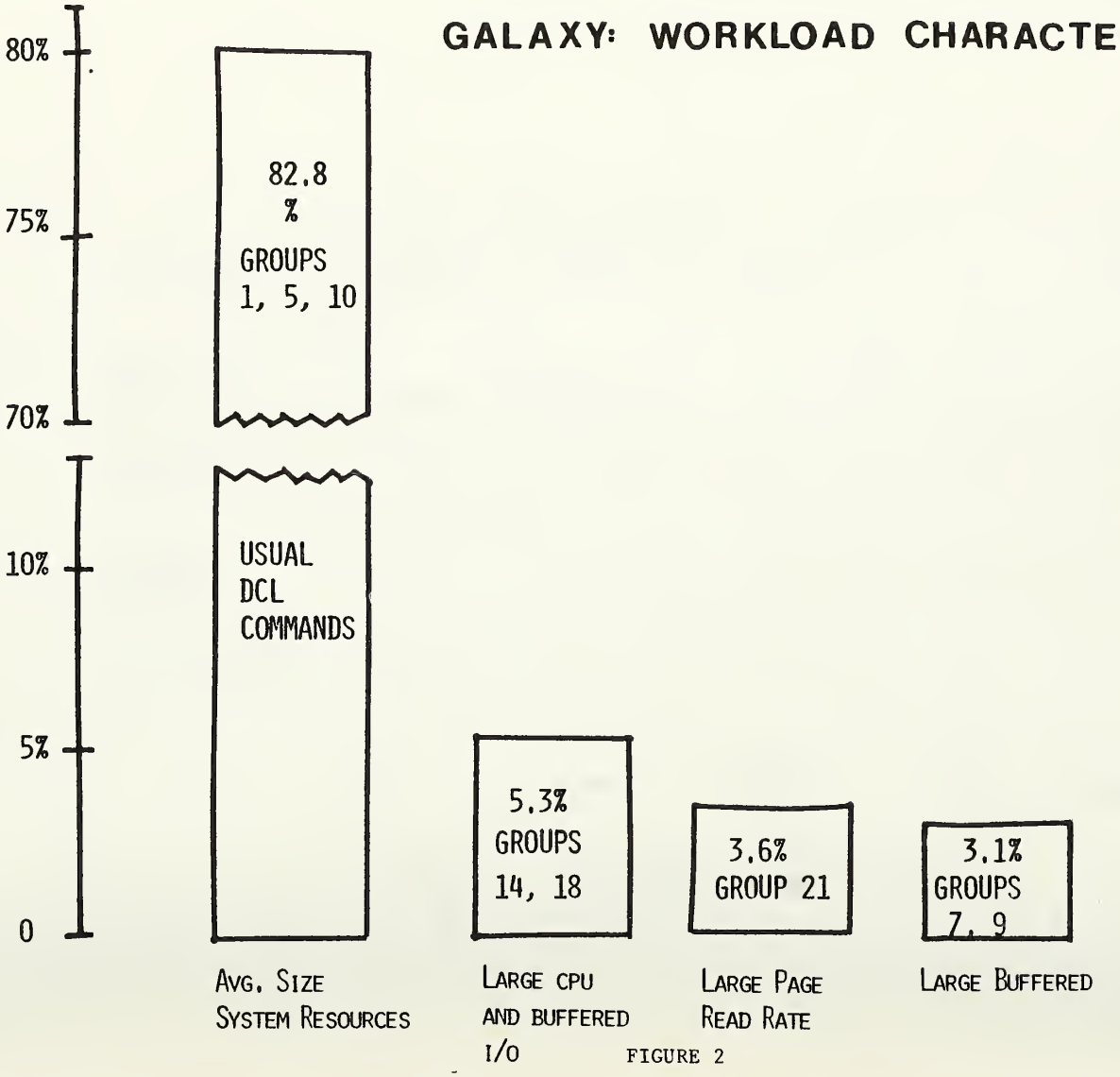


FIGURE 2

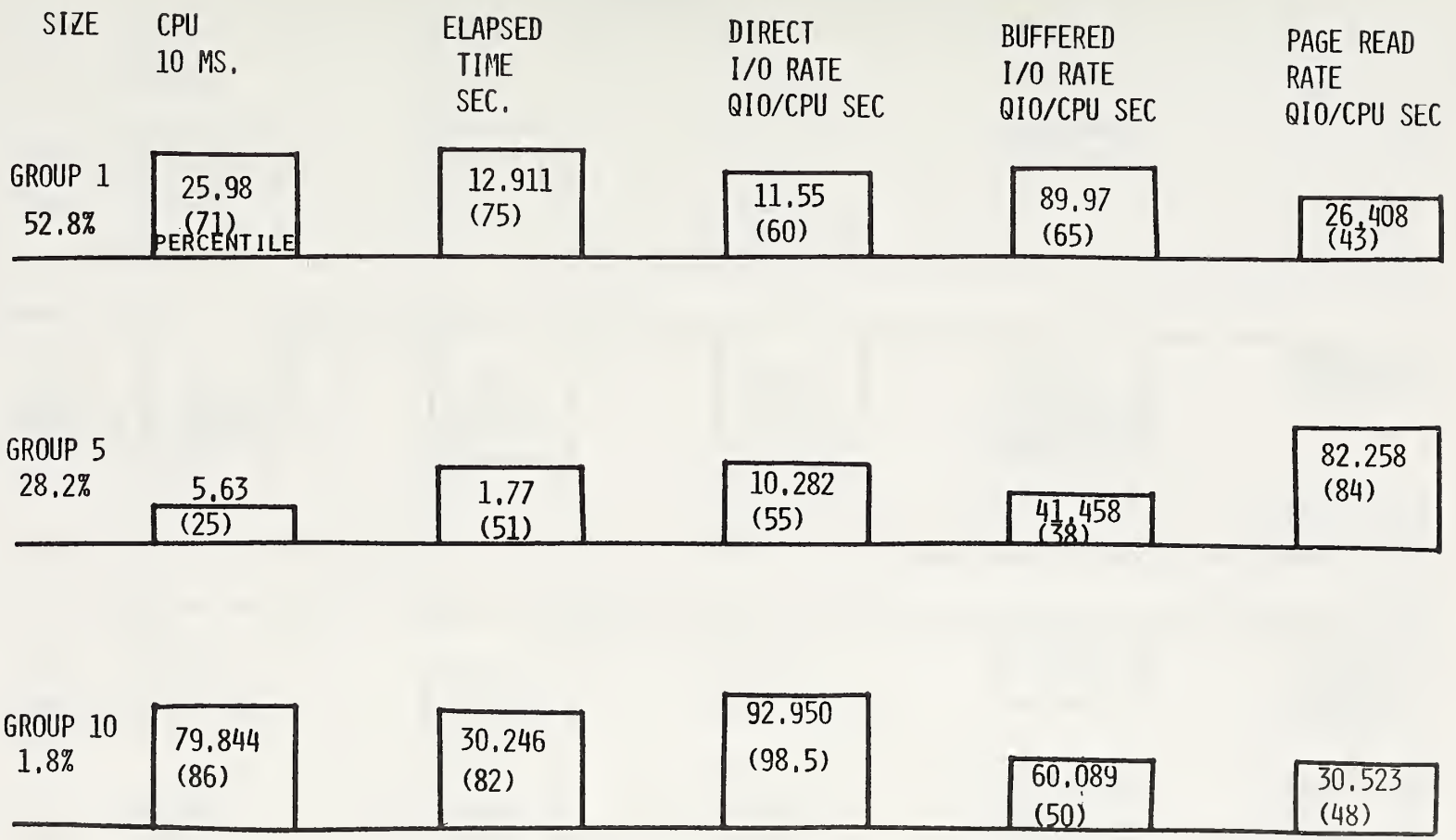

SIZE
CPU
10 MS.
ELAPSED
TIME
SEC.
DIRECT
I/O RATE
QIO/CPU SEC
BUFFERED
I/O RATE
QIO/CPU SEC
PAGE READ
RATE
QIO/CPU SEC
GROUP 1
52.8%
25.98
(71)
PERCENTILE
12.911
(75)
11.55
(60)
89.97
(65)
26.408
(43)
GROUP 5
28.2%
5.63
(25)
1.77
(51)
10.282
(55)
41.458
(38)
82.258
(84)
GROUP 10
1.8%
79.844
(86)
30.246
(82)
92.950
(98.5)
60.089
(50)
30.523
(48)

FIGURE 3

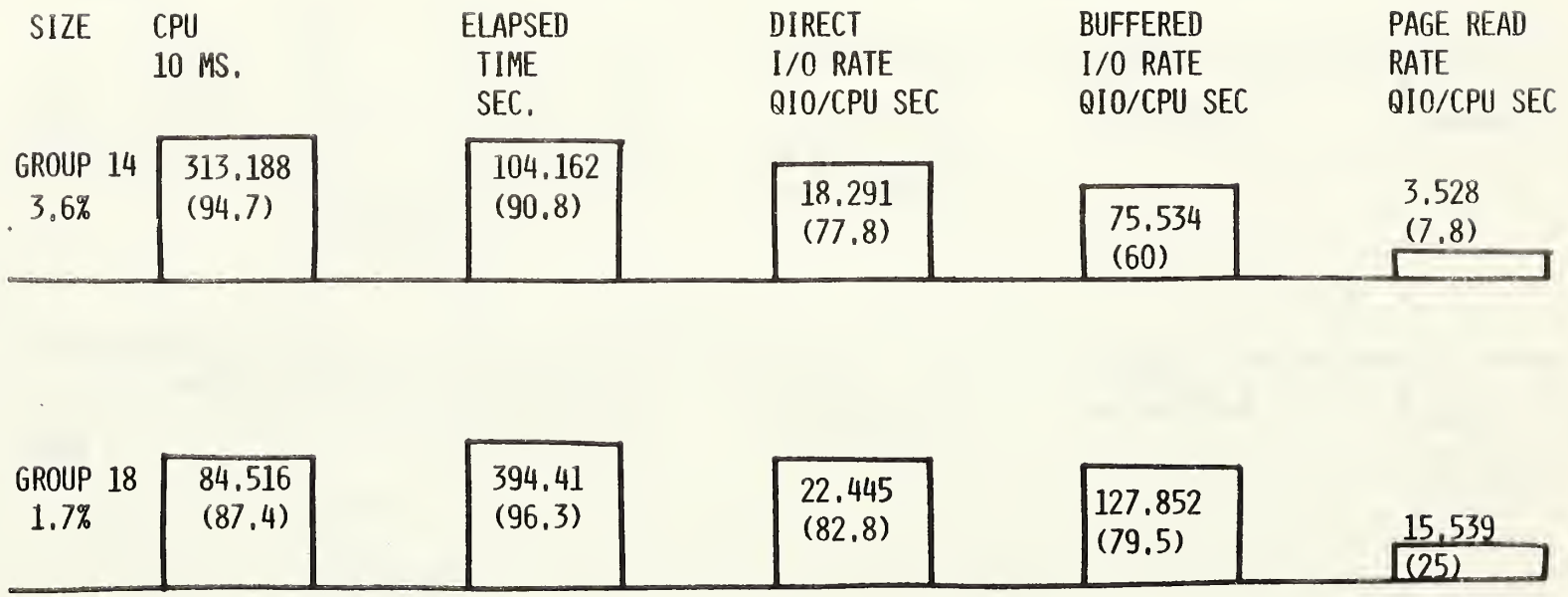

SIZE
CPU
10 MS.
ELAPSED
TIME
SEC.
DIRECT
I/O RATE
QIO/CPU SEC
BUFFERED
I/O RATE
QIO/CPU SEC
PAGE READ
RATE
QIO/CPU SEC
GROUP 14
3.6%
313,188
(94.7)
104,162
(90.8)
18,291
(77.8)
75,534
(60)
3.528
(7.8)
GROUP 18
1.7%
84,516
(87.4)
394.41
(96.3)
22,445
(82.8)
127,852
(79.5)
15,539
(25)

FIGURE 4

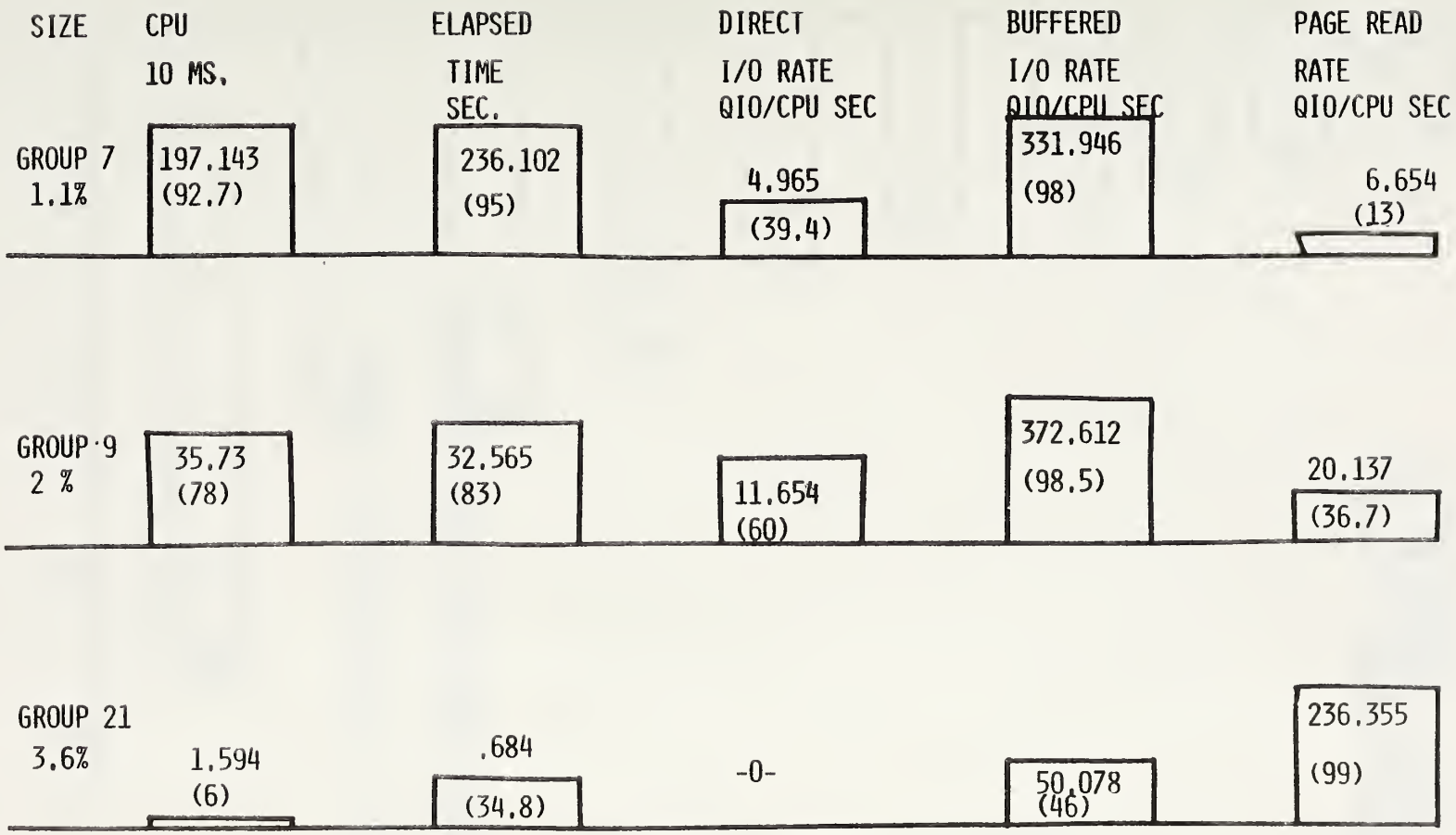

SIZE
CPU
10 MS.
ELAPSED
TIME
SEC.
DIRECT
I/O RATE
QIO/CPU SEC
BUFFERED
I/O RATE
QIO/CPU SEC
PAGE READ
RATE
QIO/CPU SEC
GROUP 7
1.1%
197.143
(92.7)
236.102
(95)
4.965
(39.4)
331.946
(98)
6.654
(13)
GROUP 9
2 %
35.73
(78)
32.565
(83)
11.654
(60)
372.612
(98.5)
20.137
(36.7)
GROUP 21
3.6%
1.594
(6)
.684
(34.8)
-0-
50.078
(46)
236.355
(99)

FIGURE 5

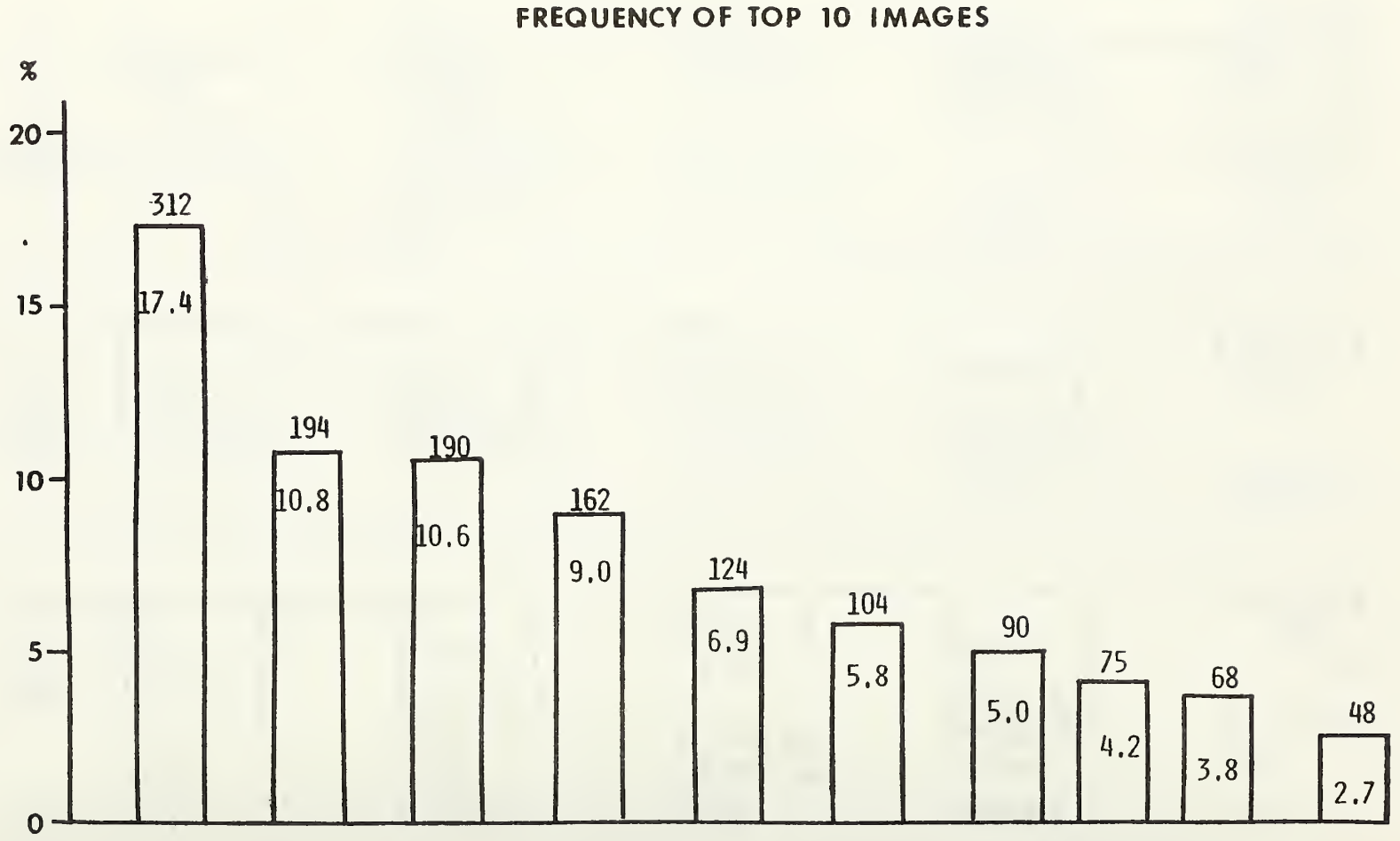

FREQUENCY OF TOP 10 IMAGES
%
20
15
10
5
0
312
17.4
194
10.8
190
10.6
162
9.0
124
6.9
104
5.8
90
5.0
75
4.2
68
3.8
48
2.7

FIGURE 6

Capacity, Disaster Recovery, and Contingency Planning

SESSION OVERVIEW

CAPACITY, DISASTER RECOVERY, and CONTINGENCY PLANNING

Chairperson: Theodore F. Gonter
U.S. General Accounting Office

Speakers: J. William Mullen
GHR Companies

Steven G. Penansky
Arthur Young & Co.

Duane Fagg
NAVDAC

ABSTRACT

At a recent computer conference a report from a capacity management/planning task force concluded that capacity needs are a business problem. Since data processing supplies the services that the business element uses to perform its functions, it concluded that requests for additional computer capacity must be made jointly between the business element and data processing. Thus, a capacity management/planning effort requires a close working relationship between the business and data processing community.

In addition, these same businesses most often depend on continuous availability of data processing services. To ensure the continuous availability of adequate services businesses develop contingency/recovery plans. These plans must provide for computer capacity considerations to ensure that sufficient capacity will be available to maintain adequate DP service.

The purpose of this session is to demonstrate the validity of the task force conclusion and the importance of capacity related considerations in the contengency/recovery planning process. The presentations and papers in this session introduce considerations in capacity management/planning, introduce the importance of capacity planning in disaster recovery planning, and offer an approach to integrating the DPI security, performance management, and contingency planning functions.

CAPACITY MANAGEMENT CONSIDERATIONS

J. William Mullen

The GHR Companies, Inc.
Destrehan, LA

Initiating an installation capacity management program required certain basic elements that must be in place in the installation to assure success of the program. These basic elements will be addressed with respect to their relationship to the business community and how this interaction should be developed in support of the capacity management/planning effort. An additional aspect of the capacity management program and function, capacity requirements in disaster recovery planning, will be outlined with respect to the aid that the capacity management function can provide to the disaster recovery planning effort.

Key words: Business case; business elements; business problem; capacity management; capacity planning; disaster recovery planning.

1. Introduction

Increased demand for computing services has given impetus to dramatic growth in many corporate data centers. Finding themselves unprepared to predict and respond to these increases in requests for computing services unable to maintain acceptable service levels for the user community and having to cope with management demands and equipment delivery schedules, installations are beginning to realize the benefits of implementing a capacity management and planning program. The obvious questions that installation personnel responsible for program implementation ask themselves are concerned with establishing a starting point for the capacity management/planning program determining if the basic elements are in place in the installation to support the program and how to evaluate program direction.

Three major topical areas should be considered in answering the above questions:

- o Environment for program implementation:
- o Elements required for the process; and
- o Communications of results to management.

Each of these major topics will be addressed in the following sections on an overview level with a final section devoted to capacity management/planning considerations in the disaster recovery plan development and how the capacity management/planning program efforts can provide aid in developing the plan.

2. Environment

Impetus for implementation of a capacity management/planning program generally stems from an environment where demands for increases in computing services from the user or business community have resulted in immeasurable and uncontrollable workload growth and a continuing decline in service levels. It is at this point where no amount of performance tuning will solve the problem, implications and consequences of the problem have been escalated to a management level most of us would like to avoid and reaction mode has become a way of life in the installation. Thus, installation personnel begin a serious attempt to establish a capacity management/planning program.

A primary consideration in program implementation is the management structure of the installation and organization and the placement of personnel responsible for performing the capacity management/planning function within the current organizational structure. Managing capacity and planning for capacity of future hardware and software components requires the availability and communication of a wide range of information. Information from lower levels within the installation concerning current configurations utilization and workload characterizations are required. Contact with middle and upper management within the business community of the organization is required to understand business strategies, business direction and to match current business demands with hardware/software component utilization and workload characterization data. Once this wide base of communications has been established, a view of hardware and software component utilization based upon business demands can be developed and an environment is created in which the installation's current capacity can be managed and future business demands on installation can be projected.

A perception of the capacity management and planning function as a business problem must be created within the installation and the business organization. This view should be based upon an economic supply and demand situation. Supply is the installation's ability to provide acceptable service at a reasonable cost. Demand is the business community's requirements for service and the time constraints imposed by these demands upon the installation. Once the capacity management/planning requirements are viewed as a business problem, personnel in the installation can begin to address the capacity management/planning function in terms of the business community as well as hardware/software component utilizations. This brings about the necessity for service level agreements.

Developing service level agreements with the business community users begins the capacity management/planning process. Service level agreements provide the target that personnel performing the capacity management/planning function must hit. From a capacity management view, the service level agreement defines current service level requirements and allows the installation to evaluate the business community's requirements for service against current deliverables. From a capacity planning view, service level agreements provide insight to short-term business requirements as well as a definition of business objectives, direction and long-range service requirements. Adjustments can be made, based upon inputs obtained from service level agreements to distribute and better manage current capacity as well as project future capacity requirements.

A basis for capacity management/planning is technical knowledge. This knowledge is normally associated with the performance analyst or software analyst function. Though expertise is required in dealing with management and planning for the installation's hardware/software component requirements, personnel performing the capacity management/planning function should have skills more aligned with management. These skills include the oral and written communications ability to translate capacity requirements from technical terms to business terms for each given level of management receiving information. The end result of this translation becomes the business case for satisfying service demands of the business community. Additionally, such skills as interviewing and negotiating will prove to

be very useful in the interaction with management within the business community structure in determining their service requirements and developing realistic service level agreements. In most installations, several people will be required to staff the capacity management/planning function to assure that the above skills are available.

The final consideration in this area is the charter of the capacity management/planning function. As stated, staff performing the function should be positioned within the organization's management structure to provide interaction with mid and upper management in the MIS as well as the business community's organization. The charter should provide freedom from day-to-day operational problem solving and still allow latitude for affecting change. Without the ability to affect change within the installation and the hardware/software components as well as the operating mode, the capacity management/planning function becomes an exercise in futility with little or no return on the investment of resources.

3. Process Elements

Technical knowledge and information is the basis of the capacity management/planning function and the raw input to the analysis and projection process. The initial information needed for the process is a complete, up-to-date inventory of the hardware/software in use in the installation and an understanding of its capabilities and resource requirements. Applications should be characterized by workload requirements and subsystems needed for their support in processing. Hardware components should be characterized by their function for support of processing and their capability and potential for growth. The inventory can thus become the base for planning growth in the installation's hardware and software components and the applications the installation must support. The inventory will also be an aid in the workload characterization process in breaking out the various business functions and systems that must be characterized.

An antecedent to the capacity management/planning program is a strong measurement base from which data on configuration component utilization can be obtained as well as resource requirements of the various business systems receiving service. Consideration must be given to the configuration and the workloads it must support to enable the installation to select those measurement tools which will provide the information required with the least impact upon the configuration. Measurement tools selected must be able to provide configuration utilization in concert with obtaining measurement data which will provide resource consumption information that can be related to the business entities receiving service. Measurement data from the tools must be in a form that will provide ease of analysis and archival for historical purposes. Currently available measurement tools (i.e. SMF, RMF) tend not to have either of these qualities in their raw form. A major consideration for the installation is implementing a central repository of information which contains those measurement elements necessary for tracking resource consumption and workload growth as well as input to workload characterization. This task has proved to be difficult for most installations. Many installations achieve some form of repository which provides configuration utilization and resource consumption growth statistics but fail to provide a mechanism for relating the utilization and consumption statistics to the various business entities receiving service.

The final consideration in the capacity management/planning process is modeling and analysis for projecting workload growth and future resource requirements. Persons undertaking the modeling and projection task will require both the low level measurement data as well as knowledge of business direction to properly apply their modeling knowledge. Without knowledge of current service requirements by the various business entities and planned business direction and strategies, the modeling process must depend entirely upon historical data which may not reflect growth patterns representative of current business direction and strategies becomes a major requirement

in the capacity management/planning process.

4. Communications

Effective communications of the results of the capacity management/planning process to management is the area where many installations fail to meet the expectations of their capacity management/planning programs. Reporting results of the program can be divided into three specific areas:

- o data processing reporting;
- o business reporting; and
- o joint reporting.

Data processing reporting revolves around configuration utilization information and component utilization at a gross and sometimes very detail level. Most installations have achieved some degree of expertise in measurement, tracking, and reporting on configuration component utilization and resource consumption growth, but do not provide business entities this information by system processed.

Business reporting revolves around reporting at the level of transactions, batch jobs, and output volumes in some form. Additional information may be provided on the contribution to component utilization by various business entities, but this is generally not done at much more than a very broad spectrum of the business community (i.e. division, group within division, section, etc.). Very few installations have the mechanisms in place to provide information at the application level.

Joint reporting involves a combination of data processing reporting and business reporting to provide management with a history and current view of configuration utilization and business entity resource consumption. Projections, based upon knowledge of business direction and strategies, can be given to management to provide a view of future configuration capacity requirements based upon business growth.

Reporting must be tempered to the various levels of management within the organization. Reports must be more concise and free from technical detail as the level of management being address moves higher up the corporate ladder. This approach to presenting a joint "business case" to higher levels of corporate management tends to strengthen the communications between DP and business management and provide impetus for cooperation in support of the capacity management/planning process.

5. Disaster Recovery Planning

Anyone involved in developing an installation disaster backup and recovery plan will atest to the difficulty in defining the processing requirements necessary at a backup site and matching these requirements to sites available for backup purposes. Several of the elements involved in the capacity management/planning process provide inputs to the disaster recovery planning process.

Inventory of applications and component processing requirements will help assure that no critical application has been overlooked in defining the applications required to continue the business processing at a backup site and will aid during backup site usage negotiations in assuring that necessary hardware/software components and options are in place at the backup site.

Resource consumption requirements of the systems that are to run at a backup site will assist the personnel responsible for site determination and selection and provide personnel at the backup site with information to judge their configuration's ability to handle the additional workload requirements.

A final aid to the disaster planning effort provided by the capacity management/planning effort is an understanding of the business direction and objectives of the organization. This understanding will help personnel involved in disaster recovery planning to identify those systems critical to continuation of the business and the

business impact loss of processing capability will have upon the organization.

6. Summary

Throughout the preceeding paragraphs, the idea of approaching the capacity management/planning process as a business problem and the involvement of mid and upper level management from the business community in the process. Effects upon measurement tool selection and reporting results of the capacity management/planning efforts to management must be approached from both the component utilization level and the business element level. This combination provides the necessary inputs to the workload characterization process, reporting process, and planning efforts for future business element capacity requirements. Data elements from the data processing environment and the business community can help achieve a viable disaster recovery plan by combining requirements for component processing and business service.

CAPACITY CONSIDERATIONS IN DISASTER RECOVERY PLANNING

Steven G. Penansky

Arthur Young & Company
Washington, DC 20036

Increasing dependence on data processing in the Federal government and private industry has resulted in growing interest in recovery plans that minimize the impact resulting from a loss to an organization's data processing capability. Although disaster recovery planning is not generally the responsibility of an organization's CPE group, capacity considerations play a significant role in the development of a cost-effective recovery plan, and CPE practitioners are being asked to participate in the development of these plans.

This paper describes and discusses several aspects of the disaster recovery planning process in which capacity considerations should be addressed. The concepts of a "minimum configuration" and "recovery windows" are discussed. Capacity considerations involved in evaluating component recovery strategies based on reciprocal agreements, equipped recovery centers, recovery shells, and multiple facilities are presented. Finally, relationships between ongoing capacity planning and recovery plan maintenance activities are examined.

Key words: Capacity planning; contingency planning; disaster recovery planning.

Most Federal agencies and organizations in the private sector rely heavily on automatic data processing (ADP) systems to achieve mission or corporate goals. Their dependence on the continuing availability of these systems makes these organizations vulnerable to disasters such as fires, flood, earthquakes, or other unexpected and undesirable events which may affect their ADP facilities. While brief interruptions in ADP service might be tolerated, most organizations would suffer considerable harm should a service interruption persist. The length of an "acceptable" service interruption varies considerably from organization to organization. However, it is clear that in most cases, just the potential of a prolonged interruption in ADP services calls for the development of disaster recovery plans to mitigate the impact of the interruptions should they occur. This paper discusses a number of capacity-related considerations which should be included when these plans are developed.

1. Background

In most ADP installations, the responsibility for disaster recovery planning has traditionally rested with operational groups. These groups, in most cases, have not fully satisfied their responsibility for protecting critical corporate or agency operations or assets. They have incorporated an increasing number of security measures designed to reduce the number of threats to which an ADP facility is subject. They have also instituted programs for the off-site storage of critical data and, in some cases, have executed letters of agreement with other

similar facilities to provide "back-up" in the event of a disaster. Actual recovery plans were developed in only a few cases and, despite these efforts, adequate protection was rarely achieved [1][1].

In recent years, data center management in both the Federal government and private industry has become increasingly aware of the importance of effective recovery plans. Reports from external auditors and examiners (e.g., reports from audits conducted by bank or insurance examiners or CPA or other firms) and guidelines issued by internal auditing groups (e.g., OMB Circular A-71 [2]) have focused management's attention on the critical importance of these plans. As a result, more concerted, corporate or agency-wide efforts have been launched to identify recovery requirements and develop a suitable recovery plan to meet these needs. It is in these more recent efforts that CPE practitioners most often have become involved.

Generally, these more recent efforts at recovery planning have taken a more realistic view of the organization's needs and have set about the development of a plan satisfying these needs. Traditionally the responsibility of operational (but, with increasing frequency, corporate security) groups, recovery planning now involves representatives from a wide range of organizational components. CPE practitioners are being asked to participate in the recovery planning process to provide much needed data on the capacity requirements which must be met by the individual recovery "strategies" which are ultimately combined to provide a framework for the full recovery plan. Although these capacity considerations are only a small part of the overall planning process, they play an important role in identifying viable strategies on which to base a plan and in ensuring that the plan continues to provide the required protection as workloads and configurations change.

2. Overview of the Disaster Recovery Planning Process

A full description of the steps involved in developing an effective recovery plan is beyond the scope of this paper. However, the various capacity considerations involved in the recovery planning process are best presented in the context of this process. The following paragraphs present an overview of this process, emphasizing those aspects of the planning process in which capacity considerations play an important role. Several of the important concepts involved in disaster recovery planning are also described. The four key phases in the recovery planning process are identified in figure 1. Section 3 discusses the capacity considerations related to each of these phases.

[1] Figures in brackets indicate the literature references at the end of this paper.

FIGURE 1

PHASED APPROACH TO PLAN DEVELOPMENT

PHASE 1 – ASSESS RECOVERY REQUIREMENTS

PHASE 2 – EVALUATE ALTERNATIVE RECOVERY STRATEGIES

PHASE 3 – PREPARE RECOVERY PLAN

PHASE 4 – REVIEW & TEST PLAN ON ONGOING BASIS

2.1 Phase 1 - Assess Recovery Requirements

The purpose of this phase is to identify the organization's recovery requirements and to determine the "recovery windows" and the "minimum configuration" which then become key ingredients in the disaster recovery planning process. The primary steps and results of Phase 1 are presented in figure 2.

The first step in this process should be the development of a series of preliminary assumptions on which the recovery requirements are to be based. In particular, these assumptions should address:

- The type of disasters which will be addressed (or ignored) by the plan,
- The availability of public utilities, transportation, and other types of support not directly controlled by the ADP facility, and
- The time (e.g., end of week, end of month, etc.) when the disaster would occur.

By identifying and documenting these assumptions, the organization can ensure that the recovery requirements are based on a common set of assumptions regarding the nature and scope of the disaster. Additional assumptions will be added throughout the planning process.

FIGURE 2

PHASE 1 — ASSESS RECOVERY REQUIREMENTS

STEPS:	RESULTS:
• ANALYZE THREATS AND EXPOSURES	• APPLICATION RECOVERY PRIORITIES
• EVALUATE LOSS POTENTIAL AS A FUNCTION OF TIME	• RECOVERY WINDOW CHARACTERISTICS
• DEFINE RECOVERY WINDOWS	• MINIMUM CONFIGURATION REQUIREMENTS
• IDENTIFY MINIMUM CONFIGURATION REQUIREMENTS	
• REVIEW WITH MANAGEMENT AND USERS	

Because the potential losses incurred by an organization will differ for each application system considered, and because, with all likelihood, the magnitude of these potential losses will change as a function of the length of the service interruption, recovery requirements should be expressed on an application-by-application basis and as a function of time. Political considerations may make it difficult to develop a precise ranking of these recovery requirements. In these cases, priorities may be assigned to classes of similar applications. At this point, a number of applications (or one or more classes) are identified as "critical applications." These applications are included in subsequent recovery planning efforts, the remainder (which might include "applications" such as systems development, performance reporting, etc.) are considered deferable until normal operations resume.

The time dependent nature of these requirements suggests that several recovery strategies may be required - each operating in one or more time frames. These time frames are termed "recovery windows" and are based on the recovery requirements identified. For each of the recovery windows identified, a "minimum configuration" must be identified. This configuration is the smallest set of:

- people;
- space, power, and other environmental preparations;
- hardware;
- software;
- data;
- documentation;
- communications (data and voice), and;
- supplies

needed to process an organization's critical applications in each recovery window. As a final step in this phase, the prioritized applications, recovery window definitions, and minimum configuration requirements are reviewed with management and users.

2.2 Phase 2 - Evaluate Alternative Recovery Strategies

During this phase, the recovery requirments determined during Phase 1 are used to analyze alternative recovery strategies and select an overall recovery approach on which the organization's recovery plan can be based. Figure 3 summarizes the key steps in this process and the results to be obtained.

Depending on the length of a service interruption, one or more recovery strategies may be needed to restore an organization's ADP capabilities in the most cost-effective manner. Multiple recovery strategies are usually required because inherent time delays, costs, or other considerations preclude the use of a single strategy throughout the duration of a disaster. For example, an organization may need 24 hours or more to

FIGURE 3

PHASE 2 — EVALUATE ALTERNATIVE RECOVERY STRATEGIES

STEPS:

- IDENTIFY VIABLE RECOVERY STRATEGIES
- EVALUATE STRATEGIES AGAINST RECOVERY REQUIREMENTS
- EVALUATE COSTS OF ACCEPTABLE STRATEGIES
- REVIEW WITH MANAGEMENT AND USERS

RESULTS:

- ALTERNATIVE RECOVERY STRATEGY EVALUATION
- COST ANALYSIS
- OVERALL RECOVERY APPROACH RECOMMENDATIONS

transfer its critical application to an alternate site (Strategy A) due to the logistical problems involved in such a transfer. As a result, manual procedures and recordkeeping (Strategy B) may be required to maintain critical functions in the hours immediately following a disaster.

The recovery strategies generally available to an organization include:

- Clerical procedures - The use of forms, logs, and other recordkeeping measures to record critical information and build a queue of work to be processed once automated processing capabilities are restored. If this strategy is to be used successfully, these forms and logs along with written procedures describing their use must be developed prior to the disaster and kept up-to-date. Because of the large number of paper records which result, clerical procedures are generally useful only for short-interval recovery windows, such as 48 hours or less.

- Formal reciprocal agreements - Agreements which guaranty that an organization struck by a disaster can share another organization's ADP facility and processing time. Though simple to establish in principle, these agreements are difficult to maintain and are unlikely to be totally reliable. The impact of such agreements on the "unaffected" organization usually limits the use of this strategy to recovery windows of one week or less.

- Recovery shells - A building equipped with the raised floor, power, cooling, fire protection, security, and other preparations needed to establish a new ADP facility but not data processing hardware or communications facilities. The viability of a strategy based on the use of a recovery shell is directly related to the hardware and communications vendors' ability to deliver replacement equipment.

- Equipped recovery centers - Complete ADP facilities, equipped with compatible hardware and communications equipment. Access to these facilities can generally be obtained within hours after a disaster is declared. However, their continued use is usually limited to between 30 and 60 days. Cost is usually the primary obstacle to the use of these facilities.

- Multiple ADP facilities - The use of two or more geographically separated facilities, the smallest of which is large enough to process the organization's critical applications. The nature of this strategy ensures the availability of a recovery facility throughout the duration of a disaster. Because the use of multiple facilities involves business decisions and commitments beyond the scope of disaster recovery, it is difficult to determine the direct costs associated with this strategy.

Once the technical, logistical, organizational, and other implications of these strategies have been identified, an evaluation on these terms is usually conducted to eliminate those strategies which are not consistent with the recovery requirements or which have unacceptable consequences in other areas. These component recovery strategies, each operating in one or more recovery windows and for some subset of the organization's critical applications, can then be combined to form one or more recovery approaches (see figure 4). The costs associated with alternative strategies and approaches is then examined and an overall recovery approach selected. The results of this evaluation are then reviewed with management and users to ensure they understand the advantages, disadvantages, costs, and any special considerations associated with each alternative and concur with the overall recovery strategy selected.

2.3 Phase 3 - Prepare Recovery Plan

Once the framework of the plan has been defined, a recovery plan development team must be established and the detailed planning, analysis, and documentation required to prepare the actual recovery plan must be performed. The primary steps and results in this phase are presented in figure 5. The specific plans and procedures to be prepared in this phase are largely a function of the particular recovery approach selected. As the final step in this phase, the plan should be reviewed with senior management, users, and ADP management to ensure that the completed plan meets the expectations of all concerned.

2.4 Phase 4 - Review and Test Plan on Ongoing Basis

During this phase, initial training for management and operational personnel should be conducted and the detailed plan developed in Phase 3 tested (see figure 6). Individuals who have direct responsibility for disaster recovery should be thoroughly versed in the plan and its execution. Those who do not should be acquainted with the policy and procedures defined in the plan.

Periodic testing of the plan is essential to ensuring that it will provide the necessary protection if and when it is needed. Tests should be conducted in a manner that evaluates the capabilities of the plan under a realistic set of conditions which could be expected during a disaster. An organization's most critical applications should be tested first, and the full range of functions required to successfully process these applications should be tested.

Maintaining the recovery plan once it has been developed is a vitally important function. As is the case in most planning efforts, the recovery plan developed in Phase 3 will be out of date by the time it is completed. The formal plan maintenance process developed in Phase 3 must be continued

FIGURE 4

COMBINATION OF STRATEGIES TO FORM OVERALL RECOVERY APPROACH

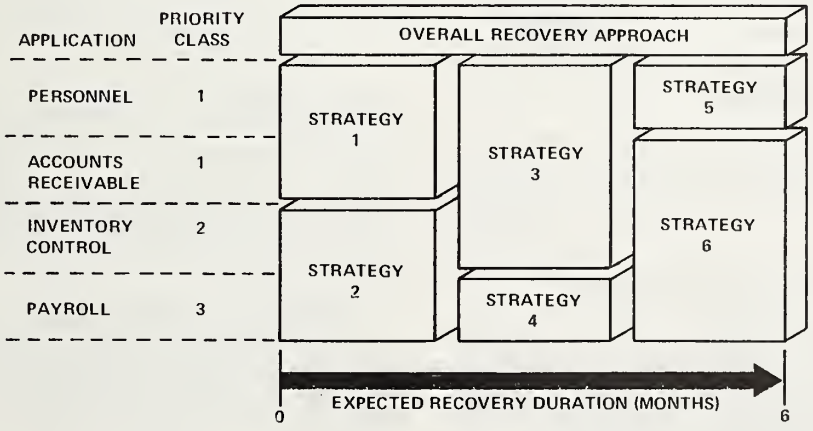


FIGURE 5

PHASE 3 — PREPARE RECOVERY PLAN

STEPS:	RESULTS:
• ESTABLISH PLAN DEVELOPMENT TEAM	• PLAN DEVELOPMENT TEAM ORGANIZATION
• PREPARE DETAILED PROJECT PLAN	• PROJECT PLAN
• CONDUCT DETAILED APPLICATION REVIEW	• APPLICATION RECOVERY REQUIREMENTS
• PREPARE PROCEDURES FOR INITIATING RECOVERY	• DESCRIPTION OF CRITICAL RESOURCES
• PREPARE PROCEDURES FOR APPLICATION PROCESSING	• DETAILED RECOVERY PROCEDURES
• FINALIZE RECOVERY TEAM ORGANIZATION AND RESPONSIBILITIES	• STAFFING AND RESPONSIBILITIES
• PREPARE PROCEDURES FOR MAINTAINING AND TESTING PLAN	• MAINTENANCE AND TESTING PROCEDURES
• FINALIZE PLAN DOCUMENTATION AND REVIEW WITH MANAGEMENT AND USERS	• APPROVED RECOVERY PLAN

throughout the life of the plan to reflect the many changes which are a regular part of data processing.

FIGURE 6

PHASE 4 — REVIEW AND TEST PLAN ON ONGOING BASIS

STEPS:	RESULTS:
• TRAIN MANAGEMENT AND OPERATIONS PERSONNEL	A DISASTER RECOVERY CAPABILITY
• CONDUCT PERIODIC TESTING	
• PERFORM ONGOING MONITORING AND MAINTENANCE	

3. Capacity Considerations

Capacity considerations are present in many aspects of the disaster recovery planning process. The following sections describe the major considerations involved in terms of each of the recovery planning process phases described above.

3.1 Phase 1 - Assess Recovery Requirements

Since Phase 1 is concerned primarily with evaluating exposure and risks and prioritizing recovery requirements, capacity considerations do not play a major role. One might argue that, at this stage, questions about:

- the capacity of an organization's ADP configuration,
- the capacity requirements of individual applications, and
- the capacity which may be required to support an organization's critical applications in a recovery environment

should not even be considered. The exposures and potential losses identified and the recovery requirements derived should be based solely on the impact of a disaster and, at this point, not on the efforts and costs required to satisfy these recovery requirements.

The early stages of Phase 1 do provide an opportunity to gather information which will be used later in this phase and in subsequent phases. This information should include:

- Resource requirements of each individual application

- Run times
- CPU and memory requirements
- DASD space requirements
- Tape drive requirements
- Communications requirements
- Printing/punching requirements
- Other special output requirements (special forms, bursting, decollating, COM, etc.)

- System and supporting software needed to process all applications
  - Operating system software
  - Special on-line systems
  - Utilities
  - Assemblers, compilers, linkage editors, etc.
  - Supporting systems (data base system(s), tape library system, etc.)
  - Others

- An inventory of all computer hardware and communications equipment
  - Type of equipment
  - Vendor
  - Model
  - Special features.

Much of this information will already be available if a comprehensive CPE program has been established.

Once the organization's recovery requirements have been defined and the recovery windows selected, the minimum configuration required to process the critical applications in each recovery window must be determined. Since many critical applications are processed on a periodic basis, this determination must be based on earlier assumptions regarding the conditions under which a disaster would occur. A "realistic worst case" scenario should be developed describing the most disadvantageous time when a disaster could reasonably occur in terms of the resource requirements of the organization's critical applications. A disaster at that time would require the heaviest use of the ADP facility and would probably result in the highest vulnerability from a corporate or agency standpoint.

Based on this scenario, the minimum configuration can be selected using the information described above or other available information sources. Each of the components described in Section 2.1 should be considered. In developing this configuration, the following should be kept in mind:

- Only the gross characteristics of the configuration need be determined; however, care should be taken to identify any special features which must be provided or any other conditions which must be met.

- Operations in a recovery environment will be difficult, at best. Few if any deviations should be made from normal operating procedures and more than an adequate supply of all critical resources should be included in the configuration.

- In particular, additional capacity should be included to account for the fact that:
  - the system used for recovery may not be tuned to the critical applications, and
  - normal processing schedules will be interrupted and additional capacity may be needed to "make up" lost time.

3.2 Phase 2 - Evaluate Alternative Recovery Strategies

Phase 2 capacity considerations center around the suitability of the alternative strategies available for disaster recovery. As defined in Section 2.2, these strategies generally include:

- Clerical procedures
- Formal reciprocal agreements
- Recovery shells
- Equipped recovery centers

- Multiple ADP facilities.

Since these strategies may be suitable for one or more groups of applications and for one or more recovery windows, the capacity considerations described should be applied to each situation for which a strategy is being considered.

3.2.1 Clerical Procedures

In the event of a disaster, clerical procedures impose a substantial burden on the personnel in user departments. These personnel, who have become accustomed to using an automated system, are suddenly forced to revert to manual recordkeeping. The difficulties they encounter stem primarily from a lack of:

- adequate staffing levels, which may have been reduced with the advent of the automated system;
- training in the manual system due to turnover, transfers, and retirement; and
- forms, logs, and procedures needed to re-establish the manual system.

Although clerical procedures are often an effective method of continuing critical functions in the period directly following a disaster, personnel capacity should be examined as a significant factor in the success of this strategy.

3.2.2 Formal Reciprocal Agreements

For disasters of short duration, formal reciprocal agreements can be effective. Some organizations can afford to share some portion of their ADP facilities with another organization for short periods of time. Difficulties are generally encountered when longer periods are involved. Because of the risks involved in assuming that only a short period of recovery will be required, and the effort needed to maintain the high degree of hardware and software compatibility a reciprocal agreement requires, these strategies are only effective in special circumstances.

Before getting involved in a formal reciprocal agreement, an organization should look carefully at its capacity requirements and the capacity available at the other site(s). It should evaluate whether it could afford to give up 30-40% of its configuration capacity and still process its workload. Does it have the time to process portions of several of the other organization's applications (each day, if necessary) to guaranty hardware and software compatibility? Does it really want to "buy into" someone else's disaster?

3.2.3 Recovery Shells

Recovery shells are relatively easy to evaluate from a space standpoint: determine the size of the site needed to house the minimum configuration(s) identified in Phase 1 including room for disk, tape, and related storage; input and output control areas; data entry; etc. Power, cooling, and several other requirements are handled the same way. If a shell looks "tight", opt for a larger one.

From a hardware and communications standpoint, the use of a shell is predicated on the vendor's ability to deliver the necessary equipment and provide the communications hook-ups needed. If possible, get vendors' commitments on replacement equipment in advance, in writing, and store them off-site. Don't forget the time required for installation and debugging. If your organization can't get a firm commitment, or if it can't afford the delays involved, use a shell for the later recovery stages and select a more reliable strategy for the earlier stages of recovery.

3.2.4 Equipped Recovery Centers

Equipped recovery centers come in sizes. The sizes are usually based on the size of the CPU in the configuration, with the number and type of DASD devices, the amount of memory, and the type of communications provided scaled proportionally. Sometimes several size options are available from the same vendor.

Selecting the right size center is relatively easy if your organization's needs fit nicely between the limits of each size class. If they don't you may be paying for more than you need, but anticipated growth over the next few years may justify the additional capacity. Otherwise you'll have to develop your plan around the features provided by the class you need now and then revise the plan a few years from now. A quick look at the features and costs of each class and your growth plans should answer the class selection question. Also remember that your plan will take six months or more to develop to the point that you could actually use the center effectively, so adjust your capacity estimates accordingly.

If your organization has some type of exceptional requirement which is beyond the scope of a size class which otherwise meets

your needs, service bureaus in the vicinity of the recovery center may be able to provide the additional capacity your organization needs. The cost may be lower than moving up a class. This approach is most applicable to COM, printing, data entry and other kinds of processing that can be done "off-line."

3.2.5 Multiple ADP Facilities

Multiple ADP facilities provide the most flexibility in recovery planning and are often the most costly strategy. However, many of these costs are not directly attributable to disaster recovery. A second ADP facility can be used in a variety of ways depending on the nature of the applications portfolio, the organization's willingness to incur rather high costs, and their willingness to undertake business ventures:

- As a ready recovery site with sufficient equipment and staff. The relatively high cost of this approach would be a direct expense to the organization. This type of redundancy might be required if recovery had to be as immediate as possible.

- As a separate but equal facility, handling approximately half of the workload and with approximately half of the total capacity. This assumes that the organization's critical applications account for less than half of the total workload.

- As a service bureau selling time for processing that could be pre-empted in the event of a disaster. In most cases, this would involve a separate business venture.

A ready site must have sufficient capacity to process the critical applications and to begin this processing within some rather stringent time constraints. This would certainly require ready capacity in all areas, including personnel. The service bureau approach imposes additional complications on an already difficult problem. Though certainly feasible, it is a decision with business and capacity considerations beyond the scope of disaster recovery.

When evaluating the suitability of each strategy and formulating an overall recovery approach, the impact of the organization's growth plans on each strategy should be considered. Each strategy should be reviewed to determine if it will still be practical and cost-effective in two to five years. The impact of each strategy on the growth plans should also be examined.

3.3 Phase 3 - Prepare Recovery Plan

During Phase 3, the detailed policies, plans and procedures to be included in the final recovery plan must be prepared. Much of this effort parallels the analysis tasks in Phase 1 only with more attention to detail. Many of the capacity considerations involved have been discussed above.

One particular procedure to be prepared in this phase involves a considerable number of capacity considerations. This procedure addresses the replacement of the ADP facility or any equipment damaged during the disaster. In the event of a partial or total loss to the ADP facility, decisions must be made about relocating the ADP facility. Similarly, if the facility can be salvaged but the configuration was destroyed, decisions must be made whether more (or less) powerful components should be acquired, and if so, which ones.

3.4 Phase 4 - Review and Test Plan on Ongoing Basis

From a capacity standpoint, Phase 4 is primarily concerned with "closing the loop" - updating the plan to reflect the changes in:

- Hardware, software, staff, etc. at the primary site

- Changes in corresponding areas at the site(s) used in the recovery plan

- Organizational policies, priorities, or practices which affect the recovery plan

- The organization's applications portfolio.

To provide the necessary updated information, close cooperation and formal interfaces must be developed between a number of facility management functions. Most of the information needed to update the recovery plan should be available from the organization's capacity planning process. Other information should be available from application system development and change control programs. Truly effective plan maintenance will ultimately require a change in the way the organization "does business," requiring new or improved interfaces between these and other ADP and business functions.

4. Conclusions

Concerns regarding the impact of processing interruptions have stimulated increased interest in effective disaster recovery plans. To ensure the effectiveness of these plans, capacity considerations must be incorporated throughout the recovery planning process. CPE practitioners should become increasingly involved in plan development and maintenance activities to ensure that recovery plans provide sufficient capacity to ensure the continued availability of an organization's critical ADP functions.

I would like to acknowledge the contribution Mr. Donald Cloud of Arthur Young & Company's Providence, Rhode Island office has made to the development of the concepts and approaches described in this paper.

References

[1] Comptroller General's Report to The Congress, "Most Federal Agencies Have Done Little Planning for ADP Disasters," U.S. General Accounting Office, December 18, 1980.

[2] Office of Management and Budget, Circular A-71, Transmittal Memorandom No. 1, "Security of Federal Automated Information Systems," July 27, 1978.

End User Productivity

SESSION OVERVIEW
END USER PRODUCTIVITY

Terry Potter

Digital Equipment Corporation
Maynard, MA

Carl Palmer pointed out in this year's proceedings Foreword that Productivity is defined as the relationship between the quantity of goods or services produced (outputs) at a given level of quality and the resources used (input). Therefore, the end user of a computer system should see higher quantity and/or quality of goods or services at the same cost if productivity increases, or at least the same quantity/quality at lower cost. Issues in measuring/reporting on productivity; techniques to increase productivity; and methods to analyze or view end user productivity are the key theme of this session.

Computers were a major boost to U.S. productivity during their early development since they were put on assignments which required little human intervention and were well structured/repetitive. Archer points out that today the applications of computers are more interactive, involving major human intervention doing work which is highly unstructured and non- repetitive. These new applications of computers are requiring fundamental changes in both hardware and software.

New hardware such as local area networks, geographically distributed systems, new database facilities, graphics and intelligent terminals, microcomputers, as well as new architectures are beginning to emerge. New types of software allowing application development without programming or, at least, more automation of application development are also emerging. Elsen's paper focuses on one aspect of this new type or style of software development.

The era of "end-user-driven" computing is now upon us. Understanding the work that the user does will be key to engineering the systems which are flexible enough to meet the productivity goals of an organization. The paper by Wilson/Mohr begins to address the issue of measuring and reporting service while allowing the user to define work in a flexible manner.

In the past, performance analysts have viewed their job as analysing the computer system and its underlying structures. In the near future, however, performance analysts will be forced to view both the computer and the personnel subsystems. Archer's paper discusses a framework for doing performance analyses at different levels of system complexity. That is, he proposes a finite state modeling framework to express both the computer and personnel subsystems. Given this framework, one can apply the typical performance analysis methods we have all become accustomed to.

SERVICE REPORTING FOR THE BOTTOM LINE

Carol B. Wilson

Fiscal Associates, Inc.
Alexandria, VA

Jeffrey M. Mohr
Chan Mei Chu

Arthur Young & Company
Washington, DC

The productivity of the end-users of an ADP installation is affected by the service being rendered by the installation, and by the anticipated service levels. Losses in productivity of the end-users can ripple outward in the form of losses in benefit to the organization from missed opportunities, etc., or in the form of increased cost to the organization to produce the same products or services. As the ADP installation becomes more critical to the success of the organization, the need to ensure the continuation of adequate service becomes more important. This paper describes the basis of an approach to user service reporting based on the fiscal aspects of ADP service. Such a system would provide the necessary information to various levels of organizational management to permit the assessment of ADP performance as it affects end-user productivity and the bottom line for the organization.

Key words: ADP installation management; cost-benefit; performance management; productivity; user service reporting.

1. Productivity and ADP Service

Productivity of the American work force has become a general issue of concern to business and government. For the past several months, hardly any issue of a management related periodical has failed to have an article on productivity. We read articles concerning the great productivity found in the Japanese industries which is no longer found in American industries. We read articles on how Theory Z of management will increase the productivity of our work force. Books have come out concerning the psychological factors of groups and norm settings, and how this affects the work flow and productivity of workers in different environments. Such concerns have also been voiced in the computer industry as the productivity of ADP personnel lags behind the technological advances made in the hardware area. In response to these concerns, the ADP industry has seen an upsurge in the development of productivity tools for both ADP professionals and non-ADP professionals using ADP as a resource.

The concept of aiding programmer productivity extends back many years. For example, the addition of the Report Writer capability to the COBOL language was an early productivity tool aimed at reducing the time and effort required to format and generate typical reports. The early efforts in the file management system area were also driven and sold for productivity increases by reducing the effort required to handle increasingly larger and more complex sets of data. Following this, products were developed; e.g., data base management systems,

which would permit a non-ADP person to access data and perform some limited formatting and manipulation of that data for reporting purposes without having to use an ADP professional as a go-between.

What has sparked this interest in productivity? Are we striving for greater productivity for the sake of greater productivity, or, is there some underlying principle that drives the pressure to maintain or to increase productivity of the work force? While there are undoubtedly sound social and psychological reasons for having people work productively, we feel that the primary interest in productivity relates to the fact that decreases in productivity decrease the benefit, or profit, to the organization. That is to say, that productivity affects the bottom line figure for an organization. This can easily be seen when we objectively look for a definition of productivity. Productivity can be defined as the amount of work performed for a given amount of dollars. Productivity will increase when more products or services are produced for a specified cost. Increasing the output of an organization - products or services - increases the benefits to the organization in the form of income, increased social benefits, etc. If the increase occurs without a corresponding increase in the cost to produce the output, the net benefit to the organization is increased.

So what does user service reporting have to do with productivity? Many ADP installations provide a valuable, possibly critical, service to the organization by providing computing resources to the organizational units to assist them in carrying out their functions. The service which the installation provides can affect the productivity of the organizational units. As the ADP installation service becomes more critical to the success of the organization, the need to ensure the continuation of adequate service becomes more important. A user service reporting system will be needed to provide the necessary information to the organizational management which will permit the assessment of ADP performance as it affects user productivity and the ability of an organization to maximize its net benefit.

During fiscal year 1981, the National Bureau of Standards supported work (under contract NB80SBCA0501) to look into the feasibility of developing a user service reporting system (USRS) which would support decisions made to maximize the net benefit to an organization. The problems associated with developing a USRS based on such a concept and the requirements which would be placed on the USRS were the major topics of the contract. In the remainder of the paper, we summarize the work of the contract and discuss the requirements of such a USRS.

2. Service and Net Benefit to the Organization

The goal of any ADP installation should be to meet the computing needs of its user community; i.e., to provide the needed service to the users. User service reporting should be the cornerstone of all computer performance management programs in that (1) reports of deviations below the required level of service should be the trigger mechanism for some type of performance improvement activity; (2) information should be provided by the reporting system to enable proper planning for the application of user workloads to the ADP installation resources to enhance productivity and maximization of net benefit. Because the requirements for service vary and the demands placed on the installation may vary, the task of managing service becomes one of global balancing and planning for the best overall benefit to the organization. It should be noted that the ADP installation itself only generates a direct benefit to the organization if it brings in outside income from "renting out" its excess capacity. Otherwise, the ADP installation's benefit to the organization can only be measured in terms of the net benefits to the organization of having the workload run on it, where the net benefit to the organization for a particular piece of workload can be defined as the difference between the benefit to the organization and the total cost of performing the work using the ADP installation resource.

How are service and net benefit related? Bennett [BENNW 81] discusses this relation by pointing out that if an ADP installation provides service to an efficient user and minimizes to a practical point the time required to successfully complete the unit of work and the required level of user effort, that the installation is providing practically maximal effective service. Further, service may be considered to have improved if the time-to-completion or user effort are reduced and such reductions lead to reduced costs to the user or increased benefits. The setting of service goals and the level of service rendered can affect the productivity of the end-users. There are no universal "best" service levels, since the levels depend upon the environment of the work force which uses the ADP resource. For example, the response time of an online system should be fairly fast for repetitive

data entry so that the system does not slow down the person entering the data. However, in some situations it has been shown that a rapid response may actually lead to greater errors, uneasiness, and lessened productivity of a terminal user.

In the section immediately below we give a general overview of the concept of maximizing net benefit to an organization. The reader is referred to the NBS special publication [MOHRJ 81a] and other references in the bibliography dealing with cost and/or benefit analysis of ADP service and the economics of computing. This discussion is included to ensure that the reader has a general notion of cost-benefit analysis and its informational requirements, since the informational requirements form the basis of the approach to a USRS. In the section following the discussion of net benefit, we examine what general requirements this approach places on a USRS.

3. Concept of Maximizing Net Benefit to the Organiztion

The crux of user service reporting and the management of service to the users lies in the identification of the computing needs of the user community. The approach taken in this paper is that analysis of computer performance should be based on the fiscal aspects of rendering ADP service to the offered user workload, and the determination of which work should be included in the offered user workload and the service it should receive be based on the net benefit of that work to the organization. The underlying assumption is that benefits and costs associated with the use of the ADP resources in support of an organizational program can be quantified. In fact, it is pointed out [BENNW 81] that if the benefit associated with a unit of work can not be determined, the decisions concerning the usefulness of the work and its relative priority with respect to other work can not be objectively determined.

The provision of a particular service or end-product by an organization is assumed to have an intrinsic value. Benefits are assumed to be the positive advantages which an organization receives as a result of supplying an end-product to another organization or individual. In the commercial sector, that value relates to profit to a business. In the Federal sector, the value may be derived from satisfying social needs of the country as expressed in the system of public laws and Congressional mandates. Qunatifying such benefits can be quite complex, and a discussion of the associated problems is definitely beyond the scope of this paper.

Although non-trivial, we will assume that the gross benefit to the organization for a particular unit of work can be determined. It remains, then, to determine the gross, or total, cost to the organization of processing that work at its ADP installation. The total cost of processing work for a project(j) can be expressed as

$$C(j) = CU(j) + CI(j) \qquad (1)$$

where C(j) is the total ADP-related cost for work for project (j)
CU(j) is the indirect costs to project(j)
CI(j) is the direct installation costs to project(j).

CU(j) relates to the service rendered by the ADP installation and would include such things as the cost of the direct users of the project (i.e., personnel who directly use the ADP resource such as programmers, data entry clerks, etc.) and delay costs (e.g., lost opportunity costs or user impact costs). CI(j) would include the computing system cost (e.g., chargeback) and auxilliary costs. It is well to note that CU(j) may be positive or negative, since

$$CU(j) = P(j) - S(j) \qquad (2)$$

where P(j) is the personnel cost
S(j) are the savings incurred as the result of productivity increases due to ADP reducing the effort.

It is at this point that we can begin to see where service, end-user productivity, and the bottom line are related. End-user productivity is affected by the service both rendered (i.e., response) and anticipated. Losses, or gains, in productivity will ultimately be reflected in costs and benefits to the organization.

Regardless of the cause and source of the ADP service degradation, the end result will be the same - some degree of impact on the user community. When the user community is unable to perform their assigned jobs due to ADP service failures, resources are wasted, user costs increase, and productivity decreases. Hence, when service rendered does not meet the required levels, delay costs may be accrued. The estimation of the more tangible costs is fairly straightforward, involving estimating the length of time a user is

non-productive while waiting for an overdue response from the ADP installation. But there are often less tangible delay costs, involving, for example, changes in work patterns of attitudes of users. An example of less tangible delay costs is given by Axelrod [AXELC 79] when he discusses the effect of a job not meeting the estimated turnaround time. The effect may range from loss time and productivity of the user as he diverts his time to a monitoring operation with respect to the system queries, to a loss of potentially profitable jobs because of lack of user confidence in the ADP installation.

Secondary delay costs may also occur as the poor service to the direct user ripples its way outward in the form of missed deadlines on projects, decisions being made using incomplete data, decisions postponed, lost opportunities for benefits, etc. It should be pointed out that resources are always wasted when service is poor, since they do not contribute to the gross benefit to the organization and could always be expended more profitably in other areas. It is not a valid argument that such costs are not real because salaries must be paid anyway, machines leased, etc., since, st some point, end-user productivity loss will translate into additional resources (e.g., manpower) requirements.

Given that the above determinations and calculations can be made, we can define as follows the net benefit to the project(j) of performing its work by using the ADP installation resources:

$$NB(j) = B(j) - C(j) \qquad (3)$$

where B(j) is the gross benefit to the organization for project(j)
C(j) is the total cost of using the ADP installation.

The equation above assumes that the installation has a particular configuration which is tuned to perform in a certain manner. When the characteristics of the installation's ability to provide service change, the NB(j) will change, because C(J) depends on the direct installation costs - CI(j) - and the performance of the installation, which affects CU(j).

For a given ADP installation capability, the net benefit to the organization as a whole for the ADP resource is defines as:

$$NB = B - C \qquad (4)$$

where B is the benefit to all projects given the installation capability
C is the cost of using the ADP installation.

Since the net benefit, NB, depends on the precise capabilities of the ADP installation and the make-up of the user workload, a set of NB(i) could be developed for various alternative installation capabilities. The proper mix of user workload and ADP installation resources could then be determined by maximizing the NB(i). It should be noted that sub-maximizations must be performed throughout the process, since the determination of whether a particular user job will become part of the workload depends on system and auxilliary costs, which in turn depend on installation capability.

In order to quantify these economic consequences, the user reporting system must be able to describe user workload at a high enough level to relate meaningfully the workload to the costs and benefits associated with the user work. In addition, methods to translate user work to installation-specific measures are needed to estimate the capacity required to support the user workload at a specified level. Such a capability would permit the costs to the installation to be weighed against the costs and benefits to the users of using the installation, all of which would lead to the maximization of net benefit to the organization. Such capabilities are not found in performance systems today. In the following section we discuss why these characteristics are important.

4. User Reporting System

4.1 Environment

The user needs the ADP installation to process his workload within certain service thresholds if the net benefit to the organization is to remain practically maximal. User service reporting enters the picture by providing the necessary information to keep the service function of the ADP installation under control - although it may not provide all the information necessary to arrive at a solution to the performance problem. Further, the user service reporting system must provide the information necessary to relate user workload to the service characteristics of a given configuration if planning is to be effective in ensuring that the needed levels of service are provided.

The goal of a user service reporting system should be to provide information with respect to ADP service to support the decisions being made by the upper management,

the project/line management, and the ADP installation management. The users need to know what service to expect and what costs will be incurred in acquiring and using the services of the ADP installation if they are to effectively control and plan their programs and projects. The ADP manager also needs information concerning the current level of service being provided to the user and the impact of the projected workload on the installation's ability to provide service. The USRS should provide an important link between the user community and the ADP installation.

Both the user and the ADP manager need service information in order to properly control the execution of their projects and programs. For example, a manager of direct users may need to make a decision regarding the addition of data entry clerks to a project if the system is not responsive enough for the current staff to perform the data entry function. Likewise, the manager of a program development group may need to make decisions regarding the assignment of multiple development tasks to an individual due to the long turnaround time of the ADP installation in processing batch work. These types of decisions are control, or operating, decisions which are made to accomodate short-term situations that exist when the ADP installation is not providing acceptable levels of service. If such situations were to occur over a sufficiently long period of time, planning decisions would be most likely be made by user managers and the ADP manager, resulting in a mutually acceptable course of action.

Decisions in the planning area require more and different information than is required for making control decisions. In particular, not only would information related to the quality of service being provided be necessary, but also information concerning the cost of providing service at different levels. Such cost information would include both the direct cost to the ADP installation of providing the required levels of service, and the indirect cost to the organization when service falls below the specified levels.

4.2 Requirements

In order for a user service reporting system to satisfactorially support both the ADP manager and the user community in effectively managing their projects, the following requirements must be met by the system:

* The ability to provide management with information necessary to make informed cost analyses in order to maximize net benefit to the organization

* The ability to provide information that is needed by the user community to control their projects and resources

* The ability to provide information that is needed by the ADP manager to control the computing resource

The first requirement is necessary if the proper planning is to be carried out to ensure that the required service level can be met by the ADP installation in the future as it processes the user workload. The last two requirements support the control, or operational, decision-making processes of the user community and the ADP installation manager.

4.2.1 Informed Cost Analysis

In the discussion above, we have noted that the computing needs of the user community should be determined by satisfying the overall goal of maximizing net benefit to the organization. The level of service provided by the ADP installation may affect both the costs and the benefits of completing the user's work functions. To be useful to a USRS report recipient, the work units for which service reports are generated should correspond to the work units of the function being performed. That is to say, the users must be given the ability to define units of work, and , thus, the levels at which the workload will be described, so that the reports are in user-oriented terms and measures. The definition of user work units will vary from installation to installation, and even from application system to application system within an organization.

Further, the full cost of ADP service must be reported, which includes both user and ADP installation costs. The service level requirements placed on the ADP installations reflect the quality of service required by the user community to successfully meet its schedules and deadlines. When the ADP installation is meeting these requirements and is available, cost incurred by projects for idle workers, missed deadlines, etc., are not attributable to the ADP resource. However, whenever the ADP installation is unable to meet the service requirements of all, or part, of its user community, there is a cost impact to projects of not having adequate service. For example, the lost time, and hence cost, associated with having data entry clerks sit idle while the computing resource is unavailable is an

additional cost incurred by the project, since the work they would have been performing must still be performed when the ADP resource becomes available. The estimation of such delay costs is not necessarily straightforward or simple. Such information, however, is essential to the user community in controlling project costs and for management in making planning decisions. In order for a USRS to satisfy this requirement, it must be able to model (calculate) the costs incurred by individual users and projects based on different levels of ADP service. When the reporting system is supporting the control function, it must be able to model delay costs based on the actual ADP service being provided. When the reporting system is supporting the planning function, it must be able to model the costs based on projected service and performance levels as estimated from workload projections and projected changes to the system configuration. The satisfaction of this objective is of great importance to the overall ability of the USRS to support the decision-making processes, particularly in the planning area.

4.2.2 Planning and Control of Projects

An ADP installation functions as a service to the user progrmmatic groups and projects of an organization. Almost all user ADP work is not an end in itself, but is a part of a larger project or program. The programs and projects have schedules and deadlines to meet, and milestones to be reached. These schedules and deadlines were projected based on the quality of service the user expected to receive from the ADP installation. The level of service was based on the function of the user project and was to help to optimize the end-user's productivity. In order for a manager to control effectively his project and meet his deadlines and schedules, he needs information concerning the service he is receiving for the ADP installation and the service that he can expect to receive in the near term.

Reporting to the user community the level of service it has received is probably not as important as projecting the quality of service that will be provided. Although it may be argued that the community already knows the level of service it is receiving from the ADP installation, and that the reporting of such informatin is not necessary, it is the inclination of users to blame the computer anytime projects are not going well, or are delayed. ADP service information would appear useful in helping a user manager isolate a problem with a project in which the service being received from the ADP installation might be a contributing factor.

The user service reporting system must be able to provide the required information concerning the projected quality of service early enough to serve as input to the decision-making process. The requirement is critical is the reporting system is to be able to support the project control function of the user. Such information is needed by the user managers to determine the adjustments that need to be made to the work schedules. This is especially important if ADP usage is on the critical path of the project. The ability of the USRS to fulfill this objective may hinge on its ability to relate service with system configuration and offered workload.

4.2.3 Control of the ADP Resource

The ADP manager has the responsibility for ensuring that the ADP installation meets the service requirements of the user community when it is processing the expected workload, and that the ADP installation is run in an efficient, effective manner. In order to carry out these functions, the ADP manager needs to know where in the system the service bottlenecks are to be found. The key to this requirement is that the USRS must be able to relate the quality of service to factors which can point to the specific areas which need improvement if the service level is to be brought within control limits. The USRS is not to take the place of a performance management system, but should be able to provide service related data to such a system. The USRS should indicate when a performance problem exists, and the performance management system should be used to diagnose the problem.

5. Conclusion

With the possible exception of ADP service bureaus or commercial timesharing services, an installation does not exist to provide ADP services, but to support an organization's mission. The service which an ADP installation provides its users can impact the productivity of the end-users and, eventually, the net benefit to the organization. Since such decisions will be made at multiple levels throughout the organization, the USRS must be capable of supplying information at the appropriate level to each report recipient. In particular, the USRS must be able to relate the service rendered to the costs of providing, or not providing, adequate service to the organizational units.

During the course of the study it was

determined that no current performance systems satisfy the requirements outlined above. The primary cause for the lack of acceptable systems is one of orientation - user versus ADP system. That is to say, the current systems are heavily ADP-system oriented and they report service in ADP units which are easy to track but which do not necessarily relate to user work functions. The USRS described above, on the other hand, must be very user-oriented if it is to fulfill its goal of assisting management in maximizing net benefit to their organization. There are three issues which must be resolved before the USRS descibed above can be developed:

* ADP system support in the definition of user work units and service characteristics

* techniques for the estimation of benefits and delay costs

* increased user involvement

The resolution of each of these issues is necessary if the USRS is to be developed. The NBS report [MOHRJ 81a] discusses the prospects for resolving these issues. Further, it discusses at length the differences in workload characterization required by this approach versus the traditional workload description. It discusses the requirements on the user community as a result of the required, new characterization. It recommends particular service measures to be reported which relate to cost-benefit calculations. The conclusion of the report is:

> The proposed approach to ADP service reporting should be useful in virtually all organizations or agencies. The approach is directed towards assisting the organization in making those decisions that will maximize net benefit by providing information describing the total costs and benefits of the level of ADP service being provided. The reports will be in terms known and understood by the user. The information in the reports should be compatible with the various types of management information presented to high level managers. The ADP service reports should be well-suited for inclusion in an overall reporting system.

References

[AXELC 79] Axelrod, C.W. Computer Effectiveness: Bridging the Management/Technology Gap. Washington, D.C. Information Resources Press, 1979.

[AXELC 80] Axelrod, C.W. The Dynamics of Computer Capacity Planning. CMG Transactions. 1980 September; 29: Section 3.

[BENNW 81] Bennett, W.D. Computer Performance: A Conceptual Framework. To be published as a NBS Special Publication, 1981.

[EDPPR 80] Fulfilling User Service Objectives. EDP Performance Review. 1980 October; 8(10): 1-7.

[FLEMI 81] Flemming, I.J. Service Level Objectives as Capacity Management Criteria. Record of the Third Annual Conference on Computer Capacity Management. 1981 April 7-9; 163-170.

[GUERJ 80] Guerrieri, J.A. Establishing True User Requirements. Small Systems. 1980 September; 8: 26-27+.

[HOWAP 80] Howard, P.C. Planning Capacity to Meet User Service Requirements. Computer Performance Evaluation CMG XI. 1980 December 2-5; Boston Massachusetts. Computer Measurement Group: 147-151.

[KLEIJ 80] Kleignen, J.P.C. Computers and Profits: Quantifying Financial Benefits of Information. Reading Massachusetts: Addison-Wesley Publishing Company, 1980.

[MOHRJ 81a] Mohr,J. and Wilson,C. Draft: A Report Describing an Approach to ADP User Service Reporting. Contract No. NB80SBCA0501, National Bureau of Standards, 1981.

[MOHRJ 81b] Mohr, J., Wilson, C., and Chan, M. Fiscal Aspects of ADP Service Management. Proceedings of CMG XII. Computer Measurement Group (to be published).

[NOLAG 79] Nolan,G.J. Standard Costing for Data Processing. Controlled Resource Management Through Computer Performance Evaluation CMG X. 1979 December 4-7; Dallas Texas. Computer Measurement Group: 12-6.

AUTOMATING THE AUTOMATIO N PROCESS

Louis C. Elsen

Evolving Computer Concepts Corp.
St. Loius, MO

The increasing demands for data automation support have created a new "frontier": A COMPUTER SYSTEM THAT CAN PROGRAM ITSELF. When one considers that $4 billion of our $6 billion Federal budget for automation support is allocated to staffing and that the introduction of new and improved technologies place increasingly difficult requirements on personnel, the need for the new "frontier" becomes apparent. The introduction of automated automation has evolved by necessity--it is a new concept whose time has come. The circumstances that brought us to this evolution stem from:

+ rapid growth in technology which makes a particular computer system virtually obsolete before it is delivered.

+ personnel which is at a crisis level. (A January MIS report predicts shortages of 50,000 in 1981.)

The result of these two "circumstances" has had a significant impact on annual cost.

+ Hardware is decreasing at 20%.
+ Manpower is increasing at 15%.
+ Software is increasing at 30%.

Today in order to operate a "typical" computer installation, an annual budget ranging from $800,000 to $1.2 million is required. This cost becomes considerably more significant when one considers the two distinct parts of each installation:

+ HARDWARE
  - -Mainframes
  - -Utilities
  - -Environment
  - -Supplies
+ STAFFING
  - -Managers
  - -Analysts
  - -Programmers
  - -Operators

The latter of the two parts, staffing, currently accounts for 70 to 80 percent of an installation's budget. This is a far cry from the cost relationships that were in existence 20 years ago. Then, an average installation was able to operate on a budget of approximately $500,000 with 65-75 percent of its total resources allocated to hardware. This impacted the growth of data automation leading to the development of computer performance management (CPM) systems which targeted efforts toward increasing hardware productivity. This philosophy and approach to CPM has continued through the years; although, the target emphasis has shifted and should now be redirected toward increasing personnel productivity.

This can be acomplished by evaluating those functions most commonly performed by data automators, reducing them to their logical components and then automating them. The end result is:

+ reduced programming requirements by elimination of menial or simple design/programming tasks;

+ increased site productivity by facilitating software generation, thereby reducing the time required to achieve a desired system objective;

+ increased programmer productivity by redirecting programming efforts to areas where automatic programming is not yet practical or achieved.

Although these "results" appear to represent an unrealistic utopia that should be reserved for a science fiction novel, this capability does exist. Basic analysis of many computer facilities revealed that the "computer's" primary function was usually nothing more than a sophisticated filing system and that the majority of the functions being "programmed" were either putting something into the "file cabinet" or getting something out. Observation of techniques employed by analysts and programmers revealed two methods for accomplishing these input/ output functions.

+ Batch: the processing of data accumulated over a period of time.

+ On-Line: the processing of data immediately with a response received as soon as possible, usually within seconds.

To continue the thrust of automating automation, one must continue analyzing data automation, further reducing it to its logical components, grouping similar functions together, analyzing how each individual component or function is performed, and finally reducing the programming logic to machine logic. We, as data automators, perform these functions daily whenever we automate a personnel, budget, or inventory system. We study the manual intensive functions being performed, reduce them to their logical components, then program the computer to do them. A computer program which is the product of the techniques I have just outlined is the Logical On-Line User's Inquiry System (LOUIS).

LOUIS is a system which writes on-line computer programs to retrieve data from the "file cabinet." It was designed to provide managers and functional users with the information they need when they need it. The programs LOUIS produces have the full range of capabilities comparable to generalized inquiry systems currently on the market. LOUIS was not designed to be the answer for everyone. It is not an update system, nor is it a batch system. The original design objectives of LOUIS were to automate approximately 60 percent of the programming requirements necessary to obtain data from an on-line data base. It performs this function well.

As alluded to earlier, programming efforts must be divided into logical categories. An attempt to automate "everything" in one huge system would probably be catastrophic. LOUIS is just one part of the entire programming scheme; incorporating the techniques employed by LOUIS and applying them to update and batch systems would yield similar results.

LOUIS is unique because it writes tailored programs optimized for the particular machine it is running on. When an individual specifies a data base name, LOUIS interrogates the data file to determine the format, type, and size of the file. It analyzes the data to determine record volume, identifies numeric data and dates, and edits them to ensure accuracy. LOUIS continues performing analysis on the information it obtains, including the types of inquiries it has executed, performing the very same functions programmers perform. The programs LOUIS writes are optimized assembly language programs. This approach achieves improved throughput and minimum system impact. To understand what (in my opinion) necessitates the use of ASSEMBLY language and why its use has diminished, I must now refer you to the past.

When computers were first designed, machine/assembly languages were the only means of "programming." Due to its tediousness and complexity, few individuals became qualified in the "art." The evolution of data automation progressed in conformance with the law of supply and demand. In order to increase the number of individuals capable of interfacing with computer, "simpler" languages were created.

+ Higher Level Languages-A language in which each instruction or statement corresponds to several machine code instruction. These languages allow programmers to write in notation with which they can become familiar.

  - COBOL
  - FORTRAN
  - BASIC
  - etc.

+ Report Writers-A processing program that can be used to generate reports from existing sets of data through parameter cards which control the flow of the program execution. Report writers usually execute in the batch mode.

- SIS
- TRS
- WWDMS (These two systems are complete data management systems which provide a capability to update the data base also.)
- TOTAL
- etc.

+ Inquiry Languages-A processing program which utilizes a simple command language whereby the interrogation of the contents of a computer's storage may be initiated at a local or remote point by use of a keyboard, touchtone pad, or other device.

  - AZ7
  - IDS QUERY
  - etc.

Although these "simpler" languages eased the loads on programmers and users, the workload on the computer was dramatically increased. For example, a program written in Assembly Language required the programmer to write 5,000 lines of code which produced 5,000 executable machine instruction. However, the same program written in COBOL or a higher level language could be written in 500 lines, but when this program is compiled and reduced to its assembly language counterpart, the executable code generated required 8,000-10,000 lines. The Report Writer, in this instance, required 50 lines of code and generated 25,000-30,000 lines of executable code. When using an Inquiry Language, 1-5 lines of code are necessary, producing 40,000-50,000 executable lines. There are three areas to be examined in this instance.

1. The Assembly Language Program is usually the ultimate in machine efficiency and productivity.

2. Programmers using higher level languages can dedicate much less time per program.

3. Through the use of inquiry languages, users can now communicate directly with the computer; thereby increasing their productivity.

This now brings us to the present. Although we are providing data automation support to managers and functional users, their need for the processing of information and the communication of that information necessary for the productive operations of an office continues to exist; and there just aren't enough data automators to program all the requirements that exist. With the introduction of such technologies as word processing, teleconferencing, and electronic mail which are becoming the vast array of the interconnected technologies available through communications networks, the need for data automators is continuing to grow. By incorporating systems such as LOUIS to perform tasks in basic system design and programming, we are able to utilize data automation personnel more effectively and thereby increase overall productivity. Management is then in a position to reduce required manpower or redirect programmer expertise to critical areas and accomplish their objectives without sacrificing internal machine efficiencies. A computer system designed to write specific programs for each user application provides:

+ System analysis and design
+ Programming
+ Elimination of program debugging time
+ Program optimization
+ Program documentation

All without the need for manual intervention.

In order to provide some validity to this presentation, here are several facts and figures obtained from evaluation of LOUIS. One installation in Denver measured an increase in productivity of ratios between 50:1 and 80:1. Eleven sites, when performing their analysis, recorded a 17:1 reduction in cost. These figures were relevant to hardware performance alone. Another installation was capable of reducing and redistributing its programmer work force by 35 individuals. In a national survey conducted by the Air Force and substantiated by the Department of Defense, savings in the quarter-billion- dollar range were achieved in just the first year.

Unbelievable? Not really. Increased productivity at each site where LOUIS was loaded was achieved because LOUIS allowed the user to request desired information in the required format when it was needed. LOUIS did not question the user's need for information. It simply analyzed the request and proceeded to write the programs necessary to obtain the answer as quickly as possible.

The philosophy expressed is shared by John J. Connell, author of an article titled "The Fallacy of Information Resource Management," appearing in Infosystems, May 1981. Mr. Connell identified four kinds of information related activities:

+ Identifying
+ Processing
+ Transporting
+ Using

He stated that the identification and use of information are the "responsibility of individual, thinking, human beings" and that the processing and transporting "can be aided by outside attention, as the successes achieved in data processing amply demonstrate."

It is my belief that the "...successes achieved in data processing..." and the roles of data automators must continue to provide users with the information desired when they need it.

However, I disagree slightly with Mr. Connell's opinion of identifying and using information. I believe these two items of information related activities can be enhanced by data automation. Through education, users can become aware of what information can be gathered, how this information can be used, and presented with new methodologies to further enhance productivity through the use of data automation.

You may ask, "Exactly where does CPM fit in this evolution?" It is my opinion that CPM should play a new role, one which incorporates management of the "total site productivity." I envision computer performance managers performing two distinct functions, utilizing automation to accomplish both.

+ The first function performed by CPM should provide users with a cost analysis of their current and projected data automation support. This service will assist users in determining whether or not the advantages of obtaining desired information is worth the expenditure.

+ The second function should concentrate on the internal workings of the site. The goal here is to achieve the most cost-effective balance between human and manchine resources.

In summary, it is important to analyze the areas I have just described because doing so can bring the realization that these jobs are currently being performed manually. It is my contention that in order to produce truly cost-effective results, the performance of these jobs must be automated!

I perceive that in the not-too-distant future, a computer system will allow a user to state his new requirement interactively at a terminal. The "computer" will ask the user pertinent questions to obtain information such as the relationships of data fields, the frequency and types of information requests, and the volume of information to be stored, retrieved, etc. The "computer" will then perform analysis as required to determine necessary hardware augmentation (if any), estimated cost, and feasibility in terms of the machine's capability. The findings will be compiled in a report to the requester and computer performance manager. It can then be reviewed to determine cessation, continuation, or modification of the plan. Once this has been acomplished, it is perfectly logical to have the "computer" go on to actually write the programs needed to implement the new procedure in a timely, efficient manner. We have reviewed and substantiated the shift in the managerial functions performed by CPM from a hardware emphasis to that of personnel. We have tracked the path of data automation and its programming evolution and shown the need for automating data automation. We have demonstrated feasibility, operability, and cost effectiveness of such.

As data automators, we can produce far-reaching, beneficial results by automating data automation. Reducing our workload, increasing central site productivity, and providing better service will significantly enhance our country's productivity and economic status. As managers of a great resource, we are bound to insure a smooth, productive, secure transition into the future providing management with the means to obtain current data necessary to efficiently manage defined areas of responsibility.

METHODOLOGY FOR ANALYZING COMPUTER USER PRODUCTIVITY

Norman P. Archer*

Digital Equipment Corporation
146 Main Street
Maynard, MA 01754

A methodology is proposed for the analysis of computer user productivity. The methodology is based upon a multi-layered level structure of the computer system in an interactive environment, where each lower level develops more detail of the interactive system. This structure enables a logical approach to the analysis of interactive systems, and productivity may be analyzed at any level, from the lowest primitive function up to the most aggregated system level. Here, considerations of productivity are considerably different in scope and direction, depending upon the level of detail. Examples are drawn from the area of text editing, which is having an increasingly strong effect on the performance of interactive systems.

Key Words: Productivity analysis; interactive systems; text editors; end user performance; performance analysis.

1. Introduction.

In the time since electronic computers were first invented and used, there has been a continuing decline in the cost per unit work accomplished by computer systems. The cost per unit time for the human resources working with the assistance of computer systems has increased steadily over the same time interval. This latter cost has been the driving force behind the ever-increasing move towards automation. In theory, the computer assists the human user to the extent that there is a net decrease in the overall cost of work done. In order to accomplish this desirable result, the computer system-human interface should be designed in such a fashion that the computer can perform more of the task with little human intervention.

This has not been difficult to do with standard clerical accounting tasks, which formed the first great wave of automated business systems. Here, in essence the computer is turned loose on a great stack of well-defined and repetitive tasks, and the results are returned to the user in some well-organized fashion at a later time. There is very little need for human intervention in such a batch-oriented environment. Hence the productivity improvement has been due to the speed at which the computer carries out its tasks, and this has shown orders of magnitude increases due to hardware evolution alone.

---

*On leave from McMaster University, Hamilton, Ontario.

But now the next wave of business systems is in the process of evolution, with the rapidly increasing usage of on-line data base systems, the automated office, timesharing computer program development and other similar uses, all sharing the common attribute that they are highly interactive systems. All carry out a variety of unstructured and non-repetitive tasks, under the close direction of the user. These interactive systems are feasible only because the system hardware costs are so low that they are no longer the important consideration they were when the first wave of business system development was in progress. Now we are faced with a different set of problems. The current concern now is with the human interface, since the user is seated in front of a terminal and a great deal of human intervention is required due to the lack of structure in the tasks the system is performing. Any improvement in the efficiency of these interactions is most beneficial, because the limiting and most expensive factor in the cost equation is the human.

A class of tasks which is consuming a rapidly increasing proportion of interactive system time includes the automated office functions such as word processing and electronic mail, as well as the more traditional task of computer program development. Of these tasks, for example, 100 percent of the time spent on a word processing system involves the use of a text editor, while measurements have shown that at least 50 percent of the commands issued during interactive program development involve text editing[1]. Thus it is very clear that user productivity is strongly affected by text editor design. Many of the human factor problems of text editors which have been addressed by other workers are reviewed in a recent article by Embley and Nagy [2]. Miller and Thomas [3] have also reviewed the more general problems encountered in the use of interactive systems. This report will discuss a structure or taxonomy for the analysis of interactive user productivity, with specific attention to the analysis of text editing interactions as examples.

2. Functional Characteristics Of Text Editors.

There are a wide variety of text editors available on modern computer and word processing systems. The functional differences among text editors are very large. However, there is a certain degree of commonality among editors intended primarily for prose editing. Similarly, there are common features among editors used primarily for computer program editing. As time has gone on, most of the more recently developed text editors have evolved towards a much more common functionality which may be used for both prose and program editing.

The line editor was the first type of text editor developed, due to the fact that the originally available terminals were hard copy terminals, which require line editors. Many varieties of this type of editor have been developed and used, primarily in systems and application programming work. The chief characteristic of the line editor is that it addresses one line of text at a time. Most line editors allow only vertical movements of the cursor, so if an operation is to be performed upon a character or string of characters on a particular line, then an editing command is keyed in which specifies the string in question, and the operation to be performed upon it. The operation may usually be extended to include a range of lines, if desired. Line editors were and are used primarily for computer program editing, since computer programs are line oriented. Some of the more specialized line editors allow line-by-line syntax checking as the program is entered, and may also allow program compilation and execution from within the editor. However, line editors do not lend themselves well to the features which are desirable in editing prose[3].

For the more general text editing tasks, including both prose and programs, a screen editor (often referred to as a character editor) appears to have several advantages over the line editor, from the user's point of view. First of all, since it is applicable to a broad range of editing tasks, it requires the user to be familiar with only one editor and one set of editing commands which may be used for any type of task. Secondly, the screen editor allows the user to view an entire page of text, and to move the cursor about on that page. The cursor is normally used to point to the text which is to be entered, modified or deleted. The cursor may be moved by keyboard command, mouse, lightpen or other pointing devices. The operation to be performed upon the text is then input to the computer by means of special function keys or, as in the case of the line editor, by keying in the instruction. The rapidly reducing costs of hardware have made the screen editor a viable tool in many interactive environments. The chief disadvantage of the screen editor compared to the line editor is that, to make it an effective tool, much higher trans-

mission rates are required from the system to the terminal and a higher price is paid in cpu time. This must be weighed against the cost of the human resources making use of the system. If the primary use of the system is computing, and machine resources are limiting, then the choice of a screen editing system is less likely. This is frequently the case in a university environment, for example. Here machine costs are usually a very strong consideration, and from the point of view of the university administration, student time used in editing text of any kind is relatively cheap. The opposite is usually the case for a firm employing clerical and professional staff for work involving text preparation, where the cost of personnel time may outweigh the cost of the additional computer resources required to support screen editing.

3. An Analysis Framework.

In order to analyze user productivity in an organized and logical fashion, the first step is to develop a framework for the analysis. One way of looking at the interactive environment is to develop a model with increasingly detailed levels of the system load in order to observe and possibly improve system performance at each of the more detailed levels. Consider the model which is illustrated in Figure 1. Since we are interested in user productivity we are naturally interested in general system characteristics in the presence of the global system workload. The global characteristics that are well known to affect user productivity are the system response time and availability. Most system performance studies concentrate on the analysis of system performance in the presence of this global load.

If we are interested in user productivity, however, we must consider the next lower level of this model, where the individual user is described by a position. This is essentially a description of the tasks which a user in this position is likely to perform, including the frequency of performing each such task type. The performance and productivity of a user in a specific position is generally given by:

a) the type and number of tasks the user must carry out,
b) the experience of the user with these tasks, and
c) the ability of the user.

Note that we are concerned here with the user's productivity while using the computer system. Clearly, we could expand the discussion to other non-computer oriented tasks if we wish, in the standard systems analysis manner. However, this is beyond the scope of this report and the discussion will be restricted to those tasks involving interactive computer system usage.

At the next lower level(the task level), more detail allows us to consider the properties of the individual tasks in which the user is involved. The characteristics of a particular task would include:

a) the type of task (for example, transcription from a typescript, editing from a marked copy, etc.),
b) the amount of time required to complete the task,
c) the expected error rate from the user/task pair,
d) the ultimate disposition of the completed task (program execution, printed copy, etc.), and
e) the experience of the user with this type of task.

The subtask level is more detailed than the task level, allowing us to address such questions as the amount of time the user spends carrying out certain activities while performing a specific task. This in turn permits more direction in performance improvement, because we may identify certain relatively primitive operations which the user performs frequently. The example shown in Figure 1 illustrates subtasks used to describe editing operations. There are three primary subtasks shown here: text entry, subtext entry, and housekeeping. The text entry subtask includes operations relevant to the direct entry of text, and related cursor movement and editing functions invoked directly by function keys. The subtext entry subtask involves those operations in which it it necessary to key in character string instructions for editing purposes, and includes function editing and cursor movement relevant to the editing of these string instructions. The housekeeping subtask includes remaining operations such as instructions for formatting printer output, filing documents, and sending electronic mail.

Each of the primary text entry subtasks includes the primitive functions of keyed text, cursor moves, and editing function commands which are at the lowest level in this structure. It is not necessary for all of the lower level functions to be common to these three subtask states. Neither must these functions be restricted to any particular subtask state. Finally, a secondary subtask state is the entry/exit

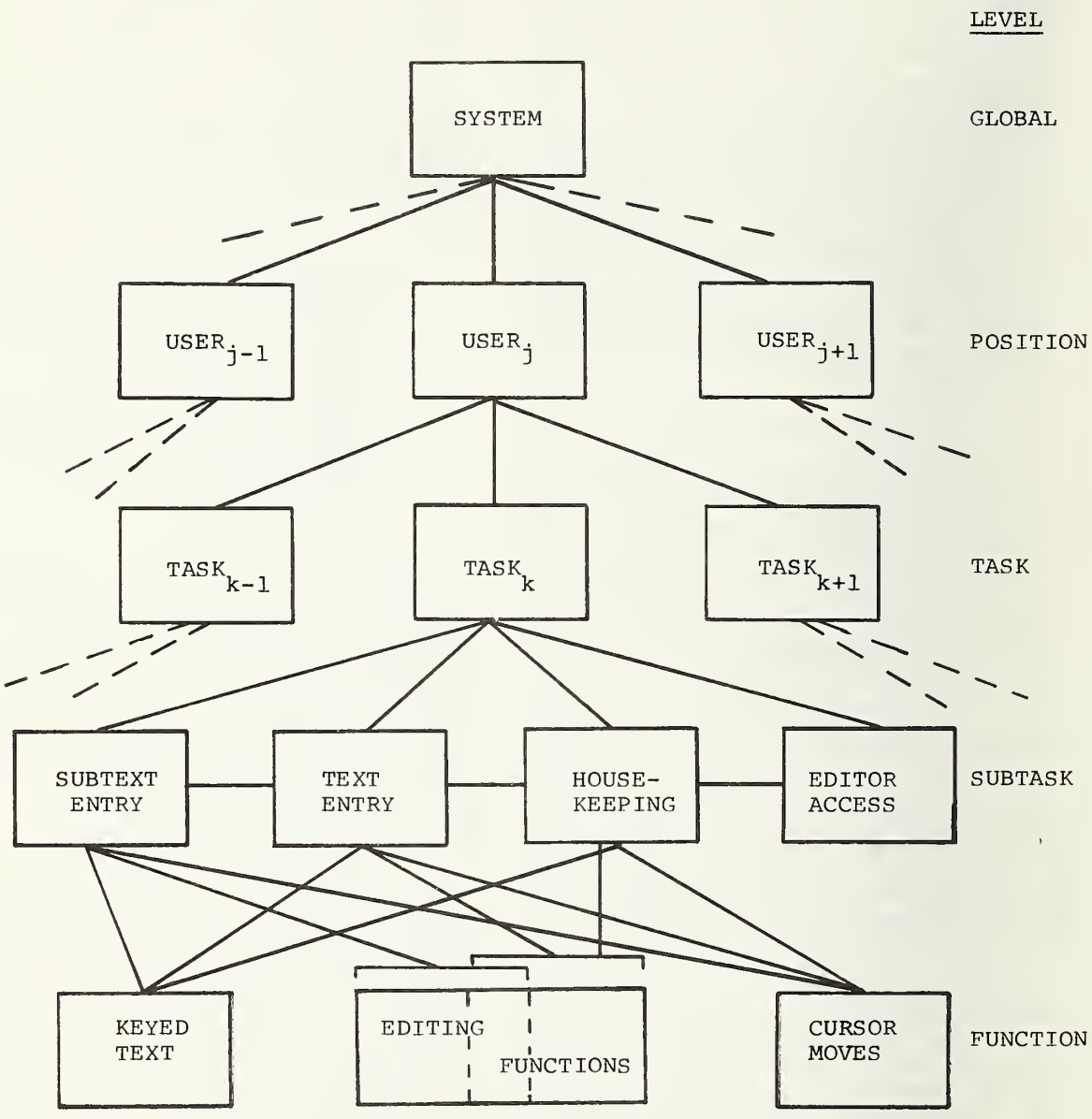


Figure 1. Text Editor Level Framework

state, which normally has little direct effect on user performance because it is entered rather infrequently.

The primitive functions which are at the lowest level in this structure include any character which may be entered into the text, any function key or keyed instruction which may carry out specific operations such as "delete word", "replace", etc., and any key or instruction which controls cursor movement. Cursor movement includes such screen editing operations as keyboard arrow keys or certain applications on light pen or mouse-driven screen editors. It may also include functions which control line editor cursors ("pointers").

Frequently used functions in text editing systems are often built into special function keys. These might include the ability to move the cursor by entities which are larger than the usual cursor movement by character, and cursor movement by word, line, paragraph or page are common. Other special function keys may be designed to allow the deletion or insertion of larger entities as well, or to perform special operations such as filing the current document.

Some frequently performed functions do not lend themselves to function key definition. For example, searching the text for a particular character string will require the user to define the desired character string. This will involve switching to what has been defined above as subtext entry mode, where the subtext is the character string in question. Similarly, a replace operation will require the definition of the string to be replaced as well as the string which replaces it. Line editors are more likely to require subtext entry of the operation to be performed. Subtext entry is not necessarily a disadvantage, because it may allow greater flexibility in user control of the editing system.

4. Productivity Analysis.

Productivity analysis considers the user time and output, and the cost of the supporting system. From the point of view of the productivity analyst, the analysis structure differs less in form than in emphasis, depending upon the level of detail upon which interest is to be focussed. For this reason, the following discussion will be directed in turn to the various levels outlined above for the analysis structure. The task level will be discussed first, because the specific task itself is a common basis which varies little, no matter which system we are considering. From here we can build to an aggregated task mix at a higher level, or we can break down the task to more detail at a lower level.

4.1 The Task Level.

To develop an understanding of the basic force which drives the evolution of interactive systems to more productive forms, it is helpful to consider various classes of interactive tasks, where the amount of structure in the tasks performed may range from very high to very low. Examples of such tasks are given below.

Degrees Of Task Structure

a) Highly Structured
   - Data entry
   - Transcribing text from document or dictation

b) Moderately Structured
   - Editing text without the aid of marked-up copy
   - Writing a computer program interactively, without the aid of notes

c) Loosely Structured
   - Writing a letter, memo or story without the aid of notes
   - Developing a project proposal from a few ideas

Highly structured tasks are on the borderline between tasks which can be completely automated and those which require some human intervention. Frequently, the human intervention is required only to transform information from one form to another. Since the ultimate bottleneck in any productivity improvement effort in this area will always be the user, technological evolution will eventually solve this problem by eliminating the human from highly structured tasks. For example, the most commonly used text entry device is the keyboard, and even the most proficient keyboard users can not improve their keying speeds above about one hundred words per minute. Added to this is the fact that a substantial number of users do not touch-type, and many get out of practice because they use a terminal infrequently. The obvious solution to this problem is to develop better text entry devices, and some progress has been made in this area. As an example, optical character readers may be used to enter printed text directly into the computer for editing and storage purposes. Some progress has been made in the area of voice text entry systems, but these are a number of years from being sufficiently

reliable and capable devices to substitute for keyboard entry.

There is less direct aid possible or even likely from computer systems in less structured environments. For example, automatic spelling verification and context editing are clearly feasible and are being used to some degree on word processing systems. But creativity will continue to be the sole province of the human in the foreseeable future. On the other hand, the computer system can be an aid to the human in improving the speed with which creative ideas can be transformed into concrete form. It is at this point where a great deal of commonality appears to exist among the writer, the artist, and the engineer. Design engineers already make use of CAD(Computer Aided Design) techniques in designing mechanical and electronic equipment and buildings, among other things. The creative, unstructured task is clearly a task in which benefits can be derived from a marriage of graphics and text editing techniques. These can combine to give the creative mind a potentially productive working tool in the form of a "professional work station". Such an interactive work station could include techniques to allow color graphics, split screen editing features and the ability to quickly generate hard copy or to set type. The writer could then have the option of integrating text with drawings and figures, thus improving the quality of the final presentation. All of these features are feasible now, but are only available as integrated packages on a few relatively expensive systems. The difficulty is, of course, in justifying these systems on the basis of productivity improvement, except in relatively specialized situations. However, technological evolution has in the past and will continue to surmount these cost obstacles.

4.2 The Global Level.

Global productivity deals with the productivity of the entire system, including the computer system and the human work force using the computer. At this level the executive priority decisions are made on the acquisition of resources to support various system applications, based presumably on productivity improvement. However, such decisions may be influenced by other factors such as prestige and the ability to attract and keep high quality employees.

Tools used for productivity analysis at this level include operations reviews for the study of human productivity, and some quantitative techniques for workflow analysis[4]. A range of measurement and analysis tools for the study and improvement of computer system performance are available. These include [5] analytic modeling, hardware and software monitoring, benchmarking, etc., around which an entire industry has grown and thrived.

For interactive systems, tradeoffs are made in the usual manner between the cost of increasing system resources and the resulting productivity improvement due to decreasing response times. It should be noted that productivity in this context is rarely quantified in an organized manner. Rather, computer system management tends to rely on the complaint level of the users as a stimulant for taking action. It is not clear that shorter response times mean higher user performance for all interactive tasks[6]. In fact, Williams and Swenson[7] suggested that system resources needed to produce short response times when not required by the user are detrimental to overall productivity; in many cases they studied, user performance was not adversely affected by longer response times. However, for most text editing tasks it is critical that response times be short, since most text editors are designed around this assumption.

4.3 The Position Level.

The analysis of the productivity of a particular position is necessarily tied in with the entire system, because of interactions among the positions in an organization. Engel et al [8] discuss the design and evolution of a prototype office automation system, which indicated a redesign of some of the lower level positions due to the delegation and shifting of certain tasks among the users. There are many issues in the automated office environment which can cause a great deal of employee concern due to the possible elimination of office jobs. Not all of these concerns are due to the complete automation of certain tasks, although this is currently an important issue in the banking industry, for example. Productivity analysis in such an environment cannot ignore the very real concerns of management and workers with both job security and the quality of the work environment.

Some issues deal with shifting of tasks to other positions, thus either reducing the prestige of the original position or eliminating it entirely. For example, consider a task which may be a source of tension in an automated office. Suppose a manager, before the introduction of automation, has dictated formal letters or memos which his or her secretary has then drafted and given to the manager for editing.

The final copy is then typed, given to the manager for signing, and finally mailed. With word processing and electronic mail, the manager now has the opportunity to do his or her own typing and/or editing, and finally to send the mail to its destination with no assistance required from the secretary. The ease of keying and making corrections to the copy is an enticement that many managers cannot resist. However, the fact that the new task may reduce the time spent on other tasks assigned to the manager's position must be considered. It is very simple to analyze the productivity improvement or reduction resulting from such a change. Suppose a manager wishes only to edit and mail the document which will be initially keyed in by his or her secretary. The decision would be in favor of the manager carrying out this task if

$$C_m < C_s$$

where C_m is the cost per task carried out by the manager, and

C_s is the cost per task carried out by the secretary

Here, $C_m = H_m \times (E_m + L_m)$

$$C_s = H_m \times E' + H_s \times (E_s + L_s)$$

where the subscripts m and s refer to the manager and secretary respectively, and

- H is the hourly rate paid,
- E is the time to correct the edited copy
- E' is the manager's time to mark up the printed copy, and
- L is the corresponding mailing time.

Analyses of this type are not appropriate for the informal short notes and reminders often communicated among managers and their employees in an automated office environment. To maintain the spontaneity, informality and speed of such communicaions, it is important that the sender interface directly with the communication system for maximum productivity.

4.4 The Subtask Level.

At the subtask level it is necessary to look at specific models of the particular class of task in question. This allows us to begin looking at issues such as software and hardware system design. Consider the finite state model shown in Figure 2, which is a model of the subtask structure level of Figure 1. The transitions among the subtasks are shown, as well as transitions among the primitive functions making up each subtask state. For a particular task, it is possible to measure the time spent by the system in each subtask state and/or function substate and thus determine how much time was spent in more productive operations. For example, keying text in the text entry subtask state is clearly the most productive operation which may be carried out, and it is more productive if no further editing is required due to entry errors or other changes. Among the least directly productive activities are moving the cursor or telling the system to perform the next editing operation. The time spent on these operations should be minimized as much as possible by improved system design.

An important issue which also directly affects the efficiency of text editing is the design of the command language with which the user communicates to the system. This was addressed recently by Ledgard, Whiteside, Singer and Seymour[9].

In order to analyze the performance of various text editing systems, we have developed a system which allows recording and time-stamping keystrokes from user terminals. Software has been developed to perform analyses on the transitions among the various states the system may occupy while the task is being carried out. The fractions appearing in Figure 2 represent the transition probabilities from each state to its possible successor states, in this case for a sample editing task being performed with a screen editor. We can also measure the time spent and the number of keystrokes transmitted while in the various model states. The following table contains a comparison of the time and keystrokes in each of the states of the subtask model, for a text entry task and an editing task respectively. Note that these results are for only one sample task in each case, so they will not be representative of such tasks on this editor. There is a wide variability among results obtained in entering and/or editing different tasks, with another source of variability being in the user performing the task.

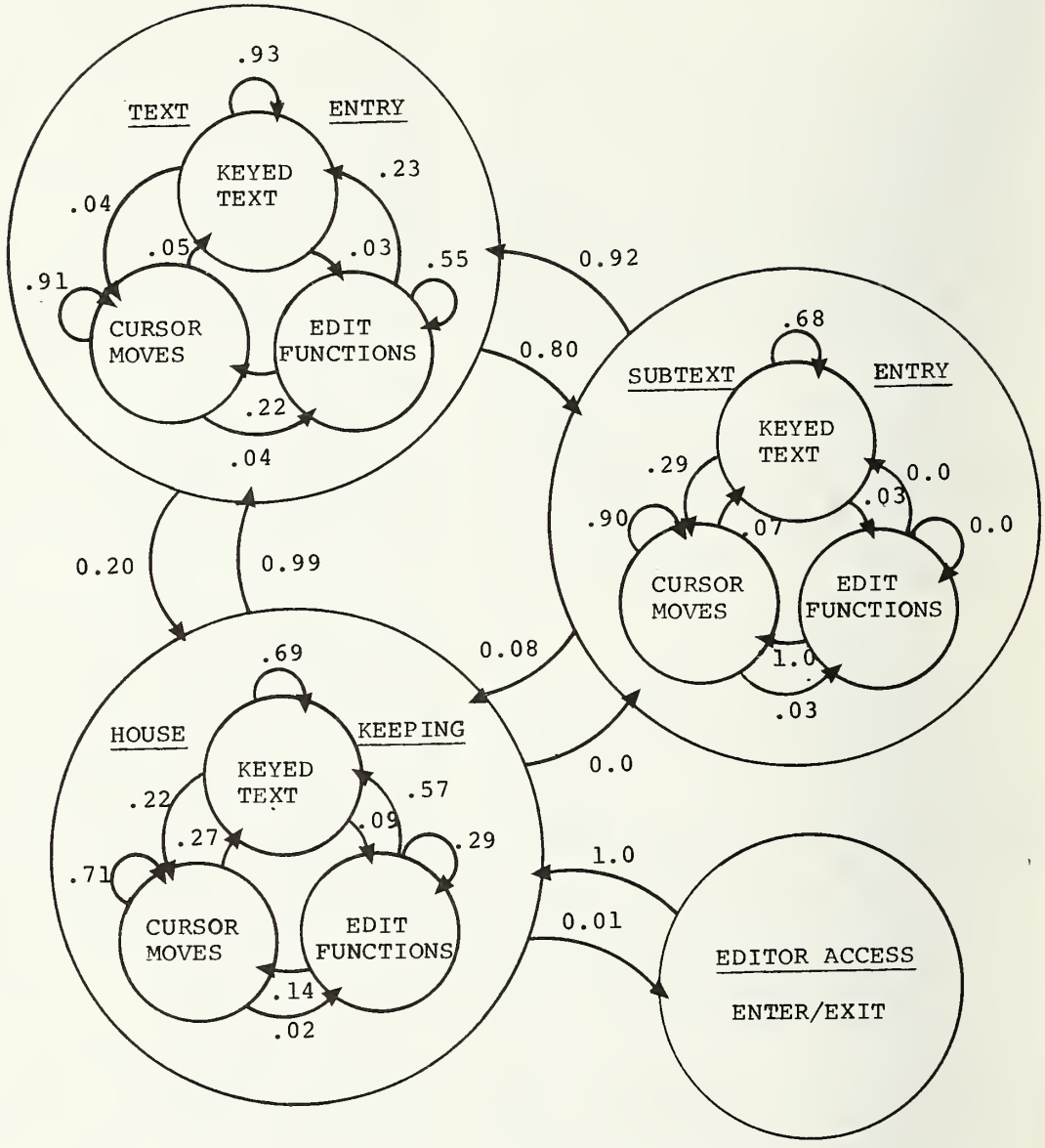


Figure 2. Subtask Interactions
For A Sample Text Editing Task

Table 1. State Time And Keystroke Analysis Of Sample Text Entry And Edit Tasks

Subtask Function	Text Entry Task Time(%)	Text Entry Task Kysk(%)	Edit Task Time(%)	Edit Task Kysk(%)
Text Entry				
Keyed Text	91.8	96.5	38.7	46.0
Edit Fns.	2.8	1.8	10.8	6.1
Cursor Moves	4.3	1.1	40.8	37.2
Subtext Entry				
Keyed Text	0.0	0.0	1.1	0.4
Edit Fns.	0.0	0.0	0.0	0.0
Cursor Moves	0.0	0.0	4.2	7.6
Housekeeping				
Keyed Text	0.3	0.3	0.8	1.3
Edit Fns.	0.0	0.0	0.5	0.2
Cursor Moves	0.8	0.3	3.1	1.2
Editor Entry/ Access	0.0	0.0	0.0	0.0

Figures given in Table 1 are rounded, so the zeros appearing in the table do not mean that no count was observed for the related function. It should be noted that, since such a high percentage of time is spent keying text for the text entry task, there may be little to be gained by system modifications for tasks of this type. Some improvement may be made by user training in keying or in correction of errors detected during keying. For the text editing task, the user usually spends a high percentage of time moving the cursor or using editing functions. Thus the editing process might be faster if modifications were made to the editing software or in keyboard design, and these are candidates for further investigation.

As mentioned above, it is important from the point of view of productivity improvement that efforts be made to maximize the amount of time actually spent keying text in the text entry mode. This is true only if the total time spent on the task also decreases. It is possible to have a system in which very little time is spent not keying text in the text entry mode, but which is so awkward to use that the total time increases substantially. This is the case with some primitive line editors which require re-entry of an entire line in order to correct an error.

A particular use of the finite state model is in the comparison of text editing system performance. By comparing state times and keystroke counts for specific tasks, it is possible to determine which system is better for that task, and also to determine why. Hammer and Rouse[10] used a Markovian model to carry out a statistical analysis of variance comparison of text editor performance in a similar situation. Most statistical tools used for such comparisons require that the finite state model be zero-order Markovian. The four state subtask model is likely to satisfy this assumption, but an analysis at the primitive function level might not, since the transition probability from one function state to another state may depend upon the previous state occupied.

4.5 The Function Level.

If a product is to be designed to perform well in a particular environment, the most important considerations, aside from the cost, involve the needs and perceptions of the ultimate user. There are many important human factors considerations in this area, such as user satisfaction, keyboard layout, and terminal and work station design. These questions have been considered elsewhere in some detail(for a review,see reference [11]).

Other considerations of importance at this level include software design, because design improvements to the software of a text entry system may improve the user's performance substantially. This may be done by reducing the time spent in activities which are not directly productive, such as cursor movement. To determine how much time is spent using each of the primitive functions, data may be gathered on representative tasks, and finite state model analysis applied at the detailed level. In this manner it is also possible to detect sequences of functions which are performed frequently and which might be automated. If cursor movement functions are found to be satisfactory, both by function design and by the placement of function keys on the keyboard, then one can turn to the design of editing features. A flood of such special features is undesirable, since a typical user has the memory capacity to learn and use only a relatively small percentage of such editing functions. Boies[12] found that only a small number of the commands available on a text editor were used. It may be preferable to have a small number of easily learned and useful features, and to invoke any little-used features through a common subtext entry function such as a function menu. A rough guide to the emphasis which should be placed on improving a particular function should be based on the frequency with which the user is expected to make use of the function. Features which could handle any well-defined tasks automatically and thus reduce the interaction time of the user are also desirable, but these are

typically heavy users of system resources such as cpu time and system memory. Functions in this category include automatic spelling verification and context editing.

In certain positions where a text editor is used a substantial fraction of the time, user-defined functions may play a role in giving greater flexibility to a text editing system. For example, if the user can define a function which will automatically generate an appropriate letterhead, with date and salutation, this can save a great deal of time. Similar comments apply to mailing lists. In the case of computer program editors, optional line-by-line syntax checkers for the particular language being used may be effective, especially for the novice. The ability to compile and execute programs from within the editor can avoid the time spent invoking and then leaving the editing system in each cycle of debugging.

5. Conclusions.

This paper has presented a framework for the evaluation of computer user productivity, with examples drawn from text editing systems, since this is an area in which a substantial impact could be made in the near future. The intention of this conceptual framework is to provide a logical form for the analysis of computer systems at any desired level of detail. We may wish to look at the global characteristics of the system itself such as the response time and availability, or we can look at the very detailed level of keystroke input. The framework is useful in helping to avoid excursions into those areas in which the analyst may not have either interest or influence for promoting change. The tools used for productivity analysis at these levels will differ widely in approach. However, the major point is that productivity improvement can be promoted at any level in the framework. A particularly useful application of this approach is in the analysis of interactive systems, the source of the examples used in this discussion.

I especially wish to acknowledge the assistance of Mr. Rollins Turner in assembling the system used to capture terminal character data for use in these studies.

References

[1] Doherty, W.J., Thompson, C.H. and Boies, S.J. An Analysis of Interactive System Usage With Respect to Software, Linguistic and Scheduling Attributes, IBM Research Report, RC 3914, 1972.

[2] Embley, D.W., and Nagy, G., Behavioral Aspects of Text Editors, Computing Surveys, V.13, 1981, pp. 33-70.

[3] Miller, L.A., and Thomas, J.C. Jr., Behavioral Issues in the Use of Interactive Systems, Int. J. Man-Machine Studies, V.9, 1977, pp. 509-536.

[4] Smith, S. A., Minimizing Processing Costs In An Automated Office System, IEEE Trans. Systems, Man and Cybernetics, V.10, 1980 pp. 232-242.

[5] Ferrari, D., Computer Systems Performance Evaluation, Prentice-Hall, Englewood Cliffs, N.J., 1978.

[6] Shneiderman, B., Human Factors Experiments In Designing Interactive Systems, Computer, V.12, 1979, pp.9-19.

[7] Williams, J. D. and Sevenson, J. S., Functional Workload Characteristics and Computer Response Time In The Design of On-Line Systems, Proceedings, Computer Performance Evaluation Users Group Thirteenth Meeting, NBS Special Publication 500-18, U. S. Dept. of Commerce, Washington, DC, 1977, pp. 3-11.

[8] Engel, G.H., Groppuso, J., Lowenstein, R.A., and Traub, W.G., An Office Communications System, IBM Systems Journal, V.18, 1979, pp. 402-431.

[9] Ledgard, H., Whiteside, J. A., Singer, A. and Seymour, W., The Natural Language of Interactive Systems, Communications of The ACM, V.23, 1980, pp. 556-563.

[10] Hammer, J.M., and Rouse, W.B., Analysis And Modeling of Freeform Text Editing Behavior, Proc. 1979 International Conf. Cybernetics And Society, 1979.

[11] Rouse, W.B., Design of Man-Computer Interfaces For On-Line Interactive Systems, Proceedings Of The IEEE, V.63, 1975, pp. 847-857.

[12] Boies, S.J., User Behaviour In An Interactive Computer System, IBM System Journal, V.13, 1974 pp. 1-18.

CPEUG81

Systems Development and Maintenance

SESSION OVERVIEW

SYSTEMS DEVELOPMENT & MAINTENANCE

Phillip C. Howard

Applied Computer Research
Phoenix, AZ 85068

This session will review several concepts that may be useful during the system development life cycle in both the development and maintenance stages. The papers address methods for simplifying that statement of user requirements, for improving the quality assurance function, and for using configuration management techniques to reduce maintenance costs.

The papers represent practical applications of several software engineering concepts which contribute to the improvement in both software quality and productivity. The specific techniques discussed in the three papers apply to different stages in the life cycle and encompass the statement of user requirements, the application of configuration management techniques, and the role of quality assurance.

DESIGN OF INFORMATION SYSTEMS USING SCENARIO-DRIVEN TECHNIQUES

W. T. Hardgrave
S. B. Salazar
E. J. Beller, III

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

This paper describes a technique for developing information systems using a scenario-driven design approach. The approach emphasizes client (that is, the user who is purchasing the system) participation in the design process. The first step is to develop a collection of "scenarios" which document the interaction between the computer and the human user. Using the scenarios, information-flow diagrams and database designs may be constructed. After the client has approved these documents, they can be used to establish disk capacity requirements and transaction rates, and finally to specify all hardware and software requirements. The primary advantage of this approach is that the scenarios provide a good indication of the ultimate usefulness and cost of the system. The client can review these documents and approve, modify, or reject the system design before any software is generated. This paper describes the scenarios, the information flow technique, and the database design approach using, as an example, a small business application.

Key Words: Database design; data dictionary; design; flowchart; information flow; information systems; interactive systems; requirements; scenarios.

1. Introduction

Designing an interactive information system requires specialized techniques that are unnecessary in the development of other types of computer systems. Human factors issues are involved, since the user community typically does not consist of computer specialists. Interactions between humans and computers must be clearly defined. The database must be described independently of any particular programs. Standard flowchart practices must be modified to capture the data flow through an interactive system.

This paper describes a technique for developing a logical design for an interactive information system. The product of this design process is a document having these components:

* Collection of scenarios
* Information flow diagrams
* Database design

Step-by-step approaches to the development of the components, which are identified below, will be presented in subsequent sections.

The term "scenario" is defined for the purpose of this paper as a detailed documentation of the interaction between a computer system and the human user working at a terminal. A scenario is essentially a hard-copy of an interactive session. This level of detail is seldom available until after the system is implemented and capable of writing output to a terminal. One goal of the scenario-driven design approach is to capture this level of detail before implementation.

The collection of scenarios is a complete description of every screen format that the terminal operator may possibly encounter. The data appearing in the scenarios should be typical of data that may actually be used in the application. Thus, this collection of scenarios completely and "by example" describes the user interface to the information system. The scenarios are discussed further in Section 2.

The information flow diagrams describe the movement of data through the system. They show where data enters the system, where it exits, where it is stored, and where it is displayed for the human user. While standard flowcharts have had widespread usage, the application of flow diagrams to interactive systems requires a somewhat different approach. A particular format and some stringent conventions are discussed in Section 3.

The database design is a complete description of the data that is to be held and permanently maintained by the system. One expects this data to have a long life span; certainly it will be longer than the execution time of any programs that manipulate it. Therefore, it is necessary to describe the data apart from any particular program or subroutine. The database design technique, discussed in section 4, is an application and extension of the "entity-relationship model" [1].

The scenario-driven technique is currently being designed as part of a project to automate the operation of a wholesale book dealer. The examples used throughout this paper are taken from that application.

There is one significant advantage of this method. There will be detailed communication between the system designer and the client (the user who has contracted to purchase the system) during the design stage, as the client will review the scenarios many times during their development. Thus, when the report on the logical design is available, the client is already familiar with the system and can be confident that it meets expected needs. There is no mystery and little chance that the client will receive a system substantially different than envisioned.

2. Scenarios

2.1 Definition

A scenario is a detailed documentation of the interaction between a computer system and the human user working at a terminal. Scenarios may be written to mimic the interaction on a line-by-line basis; a menu-by-menu basis, a screen-by-screen basis, or any other basis that can reasonably be put into the design document. For the book wholesaler system, scenarios are written on a screen-by-screen basis, that is, each complete computer-human interaction (such as "ordering", "receiving") is documented in terms of the screens sequentially viewed by the user. Each screen described here consists of a menu or a menu plus system prompts.

Most importantly, the scenario is a tool to facilitate communication between the client and the system designer. That is, the scenario is a very simple visual image that describes one part of system behavior. A comprehensive collection of scenarios can describe system behavior in enough detail to convince the client that the system will perform as envisioned.

There are some other benefits that come with the development of detailed scenarios:

* The scenarios may be included in the User's Manual as tutorial material.

* The scenarios may be incorporated into a test plan to determine system acceptance.

* The scenarios may be used as the basis for the contract between the client and the systems designer or software implementer.

The following subsections will discuss the development of screen-based scenarios and the menu format. Examples of menus from the wholesale book system are included.

2.2 Development

In a screen-based scenario design, all scenarios will have in common the top level screen, that is, the first screen that the user views. The master menu gives an overview of the aspects of the organization addressed by the information system, thereby defining the scope of the system. The designer and the client must work on the master menu until an agreement is reached and the menu satisfies both parties. During this process, the system designer and the client must learn each other's terminology and develop a common basis for further work.

After some agreement is reached on the nature of the master menu, work may proceed on the screens at the next level of detail. There is no established format for the scenarios; the screens may describe menus, user commands, or any interaction that is acceptable to both the client and the designer. This process of defining progressively lower levels of the system continues until all possible interactions have been completely specified.

There are problems involved in the management of this documentation. An exhaustive enumeration of all possible scenarios will result in a large number of screens for even a small system. As menus at different levels are redesigned, it becomes tedious to maintain consistency and verify correctness of the interfaces.

The notation used in the scenarios is not a metalanguage. The data requested from the user or printed for the user should be typical data from actual situations. This avoids any formatting problems and similar misunderstandings between the client and the systems designer.

2.3 Menu Format

The menu format is illustrated in Figures 2-1 through 2-4. Each menu (and its corresponding screen) is numbered to indicate its level, starting with the master menu as 0.0. A menu has four columns with the following headings:

* Number
* Option
* Next
* Page

Each entry in the menu represents a possible selection by the user. The "option" is the textual statement of the function that is displayed for the terminal operator. The "number" signifies the key stroke to select the corresponding option.

The "next" and "page" fields do not actually appear on the screen when the system is implemented. However, they are necessary for the writer and the reader of the logical design document in order to develop and follow the sequence of menu displays. The "next" field indicates which menu appears next if the corresponding number is selected. The "page" field gives the actual page number in the logical design document. Both of these fields are useful, albeit redundant. The page field is useful to the reader of the final design document; and the next field is useful during the development cycle, when menu numbers are available but page numbers are not.

2.4 Sample Screens and Menus

The sample screens on the following pages describe part of the "ordering" process for the book wholesaler system. The level 0.0 menu is the first menu that a terminal user would see after starting the system. "Ordering" is selected by typing "1" in response to the system's prompt "Select one:".

The "Next" field of the "ordering" entry indicates that the next menu to appear will be the 1.0 menu. After it appears, as shown in Figure 2-2, the terminal user may choose to order for a customer or order something to be stored in the inventory. In this case, a "1" is typed to signify an order for a customer.

As indicated by the "Next" field of the first entry in the 1.0 menu, the next menu to appear will be the 1.1 menu. In menu 1.1, the terminal user may choose to create a new order or to add to an existing order. In this case, the new user chooses to add to an existing order by selecting "2". The system then prompts the terminal user for the "Customer Order Group Number", abbreviated COG#. The system prints the customer identifier and the shipping address.

According to the menu of the next screen, the 1.1.2 menu, the terminal user may select to search for the title based on one of several possible inputs. In this case, the terminal user selects a search on ISBN. The system prompts for the ISBN, the International Standard Book Number, which is the unique identifier for books. After the ISBN has been entered, the system adds the book to the order group and prints a message to that effect as shown. As indicated, the screen would refresh itself to allow the operator to order another title for this customer order.

Menu 0.0

No.	Option	Next	Page
1	Ordering	1.0	2
2	Receiving	2.0	40
3	Inquiry/Update	3.0	106
4	Picking List/Invoice/PO	4.0	147
5	Accounting functions	5.0	166
6	End of day	6.0	183
7	Return to Operating System	-	-

Select one: 1

Figure 2-1 Screen 0.0

Menu 1.0

No.	Option	Next	Page
1	Order for customer	1.1	3
2	Order for inventory	1.2	26

Select one: 1

Figure 2-2 Screen 1.0

Menu 1.1

No.	Option	Next	Page
1	New order	1.1.1	4
2	Add to order	1.1.2	10

```
Select one: 2
Enter COG#: 53296

Customer Id. is: NLM
Shipping Address is: National Library of
                       Medicine
                     9600 Rockville Pike
                     Bethesda, MD  20014
```

Figure 2-3 Screen 1.1

Menu 1.1.2			
No.	Option	Next	Page
1	Search on alt. key	1.1.2.1	30
2	Search on ISBN	1.1.2	10
3	Search on series	1.1.2.3	35
4	Enter as new title	1.1.2.4	37
5	End order	0.0	1

Select one: 2
Enter ISBM: 0-13-854547

An order for Structured System Design by Gane and Sarson has been added to COG# 53296 for National Library of Medicine.

Figure 2-4 Screen 1.1.2

3. Flow Diagrams

3.1 Characteristics

Flow charts have been used for many years to describe the flow of computer programs and data. Although the diagrams used here are similar to traditional flow charts, some stringent rules for their creation and use have been imposed. These restrictions, discussed below, make the diagrams easier to read and more useful as a management tool.

As depicted in Figure 3-1, the format for the flow diagram exhibits the following characteristics:

* All flow is from left to right.
* The triggering event is the first box on the left.
* The process ends with the rightmost box (usually labeled "end").
* The left margin is partitioned by "roles"; that is, the organizational (or external) entities are listed down the left margin. One role that is usually included is "database". In this way, processes that store data in or retrieve data from the database may be documented.
* The boxes have various shapes that are assigned meanings to fit the application. This example uses trapezoidal, square, and cylindrical symbols; the meanings are explained below.

The diagram shown in Figure 3-1 is short and fits on one page; most diagrams will require several pages.

There are several advantages in using this kind of flow diagram:

* The triggering events can easily be spotted by looking on the leftmost part of the diagram.
* The involvement of a particular role can be isolated by looking along the appropriate horizontal strip.
* Time may be measured horizontally. Time intervals after triggering can be set up along the horizontal axis and charting symbols placed in the appropriate zones.
* Parallel charts may also be created. For example, in a manufacturing enviornment, a materials flow diagram can be set up parallel to the information flow diagram. The materials flow documents the flow of parts, subassemblies, etc.; the information flow documents the paperwork.

However, since a flow diagram should be generated for each possible flow sequence, there may be problems with the management of this volume of documentation.

3.2 Sample Flow Diagram

The sample flow diagram shown in Figure 3-1 depicts the flow for the part of the ordering process discussed previously. The trapezoidal boxes represent the display of a menu; the menu number is given inside the box. In cases where there is a square box underneath the trapezoid, the menu interaction requires input of a data-item by the terminal user. The name of the data-item is contained in the box. The freestanding square boxes represent processes that the system performs. If the process stores or retrieves data from the databases, this is signified by a line from the process to the appropriate database. Herein, the term "database" is used very loosely and could, in this case be used interchangeably with the term "file". The various databases are represented by the cylindrical symbols.

In Figure 3-1, the flow begins on the left when the customer prepares the order. The order then goes to the ordering department. The terminal operator calls the system into operation and the level 0.0 menu is displayed. This is signified by the trapezoid containing 0.0. The terminal operator selects option 1; this is signified by the "1" above the trapezoid. Then menu 1.0 is displayed. The terminal operator selects option 1; again this is signified by the "1" above the trapezoid. Then menu 1.1 is displayed; the terminal operator selects option 2. The box below the trapezoid indicates that the terminal operator must enter the "Customer Order Group Number", abbreviated COG#. As shown by the square box, the system retrieves the customer order group from the

"Customer Order File", abbreviated CO. Also, the system gets the "Customer Identifier", abbreviated C#, from information contained in the COG. Finally, the system retrieves the "Shipping Address", abbreviated SA and displays it along with the customer number.

The next menu to be displayed is menu 1.1.2. The terminal operator chooses to search on ISBN. The system requests the ISBN and the operator enters it. The system then searches the title file to ensure that the ISBN is valid. If the ISBN is not valid, then other menus not shown here are invoked. Finally, the title is added to the COG and the process ends.

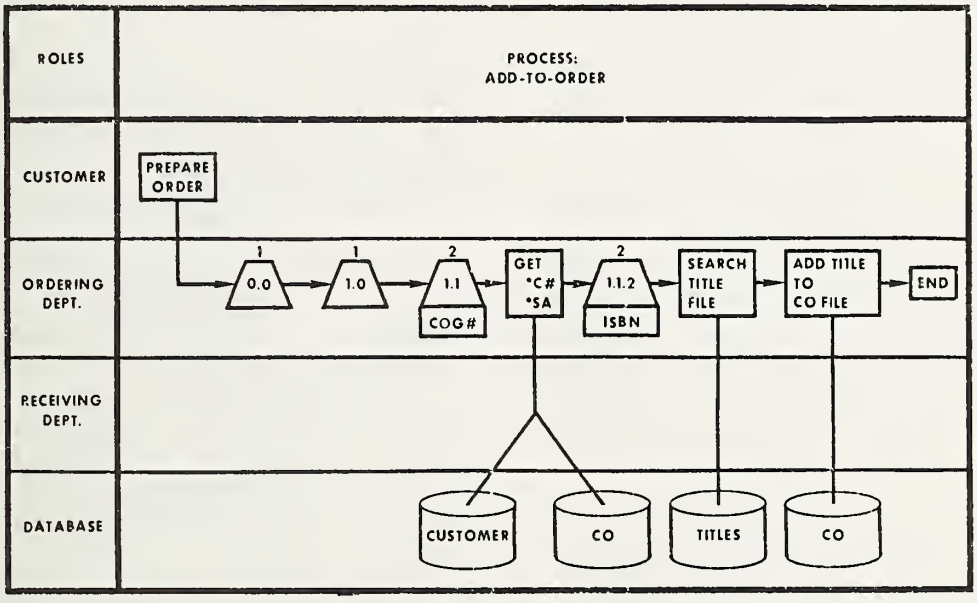


Figure 3.1: Sample Flow Diagram

4. Database Design

The database design is, roughly speaking, the format of the various data files that are to be stored permanently (usually on a disk). The actual values in the files may change as different data enters and leaves the system. We will refine this concept as we proceed.

Our approach to database design consists of several steps as described below. While it is similar to the entity-relationship approach described by Chen [1], the technique is specified in more detail and may differ somewhat from his philosophy.

The process consists of:

* Enumerating entities
* Determining the key attribute of each entity
* Defining non-key attributes for each entity
* Enumerating relationships
* Determining keys for the relationships
* Defining non-key attributes for the relationships
* Determining which relationships may also have the dual role of entities
* Assigning dual keys to dual entity/relationships

4.1 Entities

First, the designer must enumerate the entities. An entity is a concept that the designer wants the information system to recognize and manipulate. For example, in an inventory system, a part would be an entity. For some applications, there are many possible choices for entities and decisions on tradeoffs may be required. The discussion herein will center on the Wholesale Books example rather than dealing with generalities.

In the wholesale book application, the basic entities are:

* Customers
* Publishers
* Titles

Sample record diagrams giving the most important attributes of each entity are shown in Figure 4-1. In the actual database, more attributes are necessary; some are omitted for this discussion to keep it manageable.

Each entity is uniquely identifiable by a single attribute called the key (denoted by "*" in the figures). For example, a person may be uniquely identifiable by his/her social security number or a part may be uniquely identifiable by the part number.

4.2 Relationships

Next, the designer must enumerate the relationships, that is, the associations among entities. The attributes of a relationship should describe the relationship, not the individual entities, and therefore must include the keys, but no other attributes, of the entities involved.

Figure 4-2 shows some typical relationships; again these are modified from the actual application for simplicity. However, they are quite similar to an early design before the complexity required by the client was introduced. The item-order relationship relates customers to titles. A customer (CID) orders a title (ISBN) in some quantity (QTY). The title/publisher relationship relates titles to publishers. A publisher (PID) publishes a title (ISBN).

Our example is somewhat more complex. The item-order relationship needs to be grouped to reflect which books appear on an incoming order. Also, the item orders need to be grouped to reflect which items are included on a single order to the publisher. Therefore, the item-order relationship needs to be referenced as an entity in other relationships. This is accomplished by adding an attribute, Customer Order Group (COG#) which becomes the key of the augmented item-order dual entity/relationship (as shown in Figure 4-3, with the key denoted by "**").

4.3 Data Dictionary

The purpose of the data dictionary is to provide a written description of each attribute that is referenced in the database (in both entities and relationships). The data dictionary can be used as a basis for communication among the various people working on the questions that arise during the design process concerning the purpose and meaning of various data-items. Whenever the database is reorganized, the data dictionary is updated to reflect the new organization. Thus, the data dictionary is an important tool for managing and controlling the system design process.

TITLES				
ISBN*	Title	Author	Cost	Price

CUSTOMER		
CID*	C-Name	C-Address

PUBLISHER		
PID*	P-Name	P-Address

Figure 4-1 Entities

ITEM-ORDERS		
CID*	ISBN*	Qty

TITLE/PUBLISHER	
ISBN*	PID*

Figure 4-2 Relationships

ITEM-ORDERS			
COG#**	CID*	ISBN*	QTY

Figure 4-3 Dual Entity/Relationship

Although the creation and maintenance of a dictionary can be an expensive undertaking, it is possible to scale the complexity to fit the application. For example, the dictionary may contain many attributes for each data-item or as few as two; it may be managed manually or with a computerized system.

Because this approach is intended for a small organization with limited resources, the data dictionary described here is as simple as possible and may be managed manually at a small cost. As depicted in Figure 4-4, the dictionary is a table with two columns: data-item and narrative. Each

entry in the table describes one data-item, or attribute, that is defined in the system. The column labeled "data-item" contains the names of the data-items, while the column labeled "narrative" contains descriptions of the data-items meaningful to the user. The usefulness of the data dictionary (as well as its complexity and cost) could be increased by adding more information to each entry. For example, a column "entity" could be added to keep track of the entities in which each data-item appears.

DATA DICTIONARY	
DATA-ITEM	NARRATIVE
ISBN	International Standard Book Number
Title	Title of book
Author	Author of book
Cost	Publisher's price for book
Price	Our normal price for this book
CID	Unique identifier for customer
C-Name	Name of Customer
C-Address	Address of Customer
PID	Unique identifier for publisher
P-Name	Name of publisher
P-Address	Address of publisher
Qty	Quantity: Number of this ISBN ordered by this CID
COG#	Customer Order Group Number: Unique identifier for Purchase Order

Figure 4-4 Data Dictionary Table

5. Concluding Remarks

The information system design technique presented in this paper is based on a surprisingly simple but rarely-used idea--that the systems analyst build a mock-up of the system for review by the client before implementation begins. The method of constructing the mock-up, described herein, is the development of a collection of "scenarios," the possible interactions between the computerized information system and the human user. The client and the system designer take part in a number of iterations of reviewing and revising scenarios, while the information flow and the database design proceed in parallel. Approval of a collection of scenarios by the client is the first milestone in the development of the system.

References

[1] Chen, Peter, "The Entity-Relationship Model--Toward a Unified View of Data", ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, pp. 9-36.

[2] Gane, Chris and Sarson, Trish, Structured Systems Analysis: Tools and Techniques Prentice-Hall, Englewood Cliffs, NJ, 1979, 241 p.

CONTAINING THE COST OF SOFTWARE MAINTENANCE TESTING -AN EXAMINATION OF THE QUALITY ASSURANCE ROLE WITHIN U.S. ARMY COMPUTER SYSTEMS COMMAND

MAJ Steven R. Edwards

Quality Assurance Directorate, US Army Computer Systems Command
Ft. Belvoir, VA 22310

This paper provides a brief introduction to the US Army Computer Systems Command (USACSC) to include its various elements and their respective functions. The previous (prior to 1 Sep 80) requirements for software maintenance testing within the Command are examined, with the greatest emphasis being placed on the specific allocation of the Quality Assurance Directorate's resources (manpower and computer) necessary to fulfill these requirements. Key points in this portion include an explanation of Developmental Center Testing (DCT), Environmental Testing (ENT) as an independent, third-party test, and Field Validation Testing (FVT). The evolution of third party testing within the Command is also discussed. Major points envisioned in the revised testing procedures which the Command and QAD are implementing are detailed including a discussion of the advantages and disadvantages of independent testing as perceived by various members of the academic community as well as by the individuals directly involved with this independent testing within USACSC. Conclusions reached as a result of weighing these advantages and disadvantages and their ultimate impact on the revised testing procedures are also examined. The underlying point is the anticipated increase in effectiveness of the Quality Assurance program obtained from the readjustment of resources, the gradual phasing out of the Environmental Test, and the subsequent increase in "up front" QAD involvement. Concluding comments encompass an overview of the methodology being developed to gather and evaluate feedback data on the success and/or shortcomings of the revised procedures. This feedback data includes reports of problems in fielded systems, especially immediately subsequent to the release of maintenance changes to this software, as well as requested changes to fielded systems.

Key words: Developmental Center Test, Environmental Test, Field Validation Test, Independent Testing, Quality Assurance Program, Software Change Package, Software Maintenance Testing.

1. Introduction

This paper examines the issues of software maintenance testing as a part of the quality assurance mission within the United States Army Computer Systems Command (USACSC). Such an examination is particularly relevant at this time, since a significant change in the procedures is in progress. Thus, it is logical to document the software maintenance testing requirements within USACSC prior to September 1, 1980, discuss the perceived need for change in these procedures, and then detail the revisions underway. Since maintenance testing is an integral part of the overall quality assurance effort within the Command, any change in this part necessitates an in-depth review of the role of the quality assurance element in general. This

consideration is also addressed in the subsequent discussion.

In order to provide the desired setting for a discussion of the specific topics described above, a brief overview of USACSC is apropros. This introduction to the command focuses on its mission, organization, and functions in the role of a central design agency responsible for developing automated systems for the Army. The mission of the Command is shown in figure 1.

USACSC MISSION

- SERVE AS THE PRINCIPAL ARMY DEVELOPER OF MULTICOMMAND ADP SYSTEMS.
- DESIGN, TEST, INSTALL AND MAINTAIN ADP SYSTEMS.
- DIRECT THE ARMY ADP STANDARDIZATION PROGRAMS.
- PROVIDE TECHNICAL ASSISTANCE TO HQDA STAFF AGENCIES AND MAJOR ARMY COMMANDS.
- CONDUCT SOFTWARE RESEARCH PROGRAMS.

Figure 1. USACSC Mission

Multicommand ADP systems are in support of functional applications which are similar in two or more major Army Commands. Figure 2 shows how the various elements within the Army interact with USACSC in support of this mission. The Assistant Chief of Staff for Automation and Communications is responsible for establishing overall objectives, policies, and procedures for Army Staff direction and guidance for the design, development, installation, and maintenance of Army management information systems. (A planned reorganization within the Army Staff will alter this element; nevertheless, USACSC will continue to receive Army Staff guidance and direction.)

Within their respective areas of responsibility, the Army Staff agencies formulate statements of requirements for new systems, or changes to existing systems, which are then forwarded to USACSC. As the principal Army developer for Standard Army Multicommand Management Information Systems (STAMMIS), USACSC receives the approved requirements and undertakes systems design or change. Additionally, USACSC

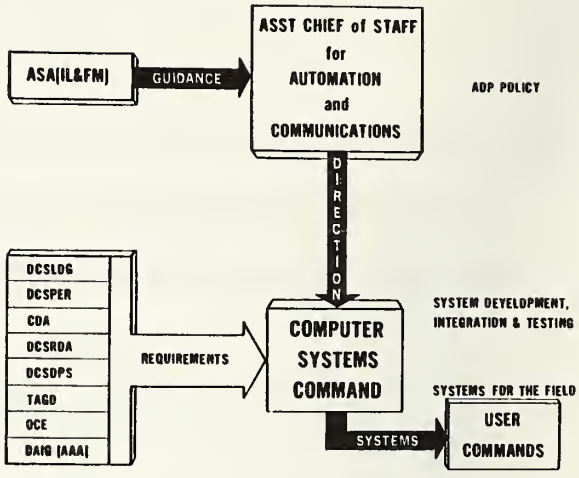


Figure 2. Army Interactions With USACSC

provides technical support to user commands, both in the US and overseas. The authorized strength of the Command includes 382 military and 1102 civilians scattered throughout nine geographic locations. Systems interfaces are a major concern of the command. Figure 3 shows a typical array of interfaces for one of the standard systems. Documentation and configuration management of interface specifications are extremely difficult. An element that makes this control even more complex is the fact that different systems developers may be involved. Hence, a developer could easily make changes in specifications to one system without anticipating the impact these changes could have on other systems. To preclude problems in this area, interface control is exercised at Command level. This centralization within a single central design agency permits the coordination and management of interface requirements between systems controlled by different proponent agencies.

This paper focuses on the Command's Quality Assurance Directorate (QAD) and its interactions with three of the Command's software developers: Logistic Systems Directorate, Personnel and Force Accounting Systems Directorate and Financial Systems Directorate. These Assigned Systems Developers (ASD) design, test, install, and maintain over forty standard systems in response to Department of the Army Proponent Agencies (PA) approved requirements. The STAMMIS are operational at

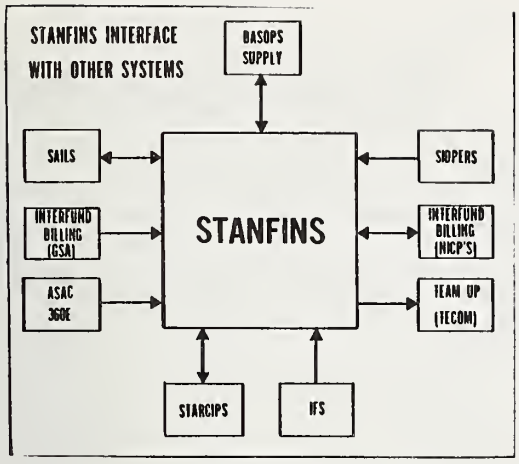


Figure 3. STANFINS Interface With Other Systems

approximately 150 Army Data Processing Installations world-wide. Some of the functional requirements that are satisfied by the standard systems developed by USACSC are portrayed in figure 4.

FINANCE	PERSONNEL	LOGISTICS
INSTALLATION ACCOUNTING	STRENGTH ACCOUNTING:	SUPPLY MANAGEMENT
INVENTORY ACCOUNTING	ACTIVE ARMY	MAINTENANCE REPORTING
CIVILIAN PAY	RESERVES	COMMISSARIES
DATA INQUIRY	RETIRED	PORT OPERATIONS
R&D BUDGETARY CONTROL	CIVILIAN	INSTALLATION SUPPORT
	FORCE DEVELOPMENT	AMMUNITION
	MILITARY POLICE	

Figure 4. Functional Systems Operational

2. Software Maintenance Testing

Having provided this broad description of USACSC, it is now appropriate to delve more deeply into its software maintenance testing. Maintenance testing is the testing of approved software changes which result from systems enhancements proposed by the PA and Army activities currently processing the system. The assigned developer, in conjunction with the proponent, programs the necessary changes and assembles the altered software into a systems change package (SCP). Formal test condition requirements, which specify the requirements necessary to test the change, are prepared by the proponent for functional changes and the developer for technical changes. Maintenance testing is composed of three levels of evaluation conducted by the system developer, followed by an independent third party test conducted by the Quality Assurance Directorate. At the conclusion of these tests, the systems change package is subjected to its final test at a field site where it is evaluated by representatives of the major Army command user community (with Quality Assurance, Proponent Agency, and System Developer personnel on site). These tests are further defined in USACSC regulations as follows [1]:

Developmental Center Test Level I (DCT 1) - A test which occurs after design, coding, desk checking, and successful compilation of a new or changed program has occurred and which insures that the program operates efficiently. Primary objectives of the DCT I are:

- Identify and correct all program errors.

- Exercise program decision and logic flow affected by change to demonstrate technical correctness of program.

- Exercise program functions affected by change to demonstrate functional correctness of the program.

- Assure compliance with programing and documentation standards.

The DCT 1 is normally performed by the programer who conducts the test utilizing developer checklists (no formal test plan). The programer examines the developers existing test data base and prepares additional test data as necessary to test all data types which have been affected by change. At the conclusion of the test, the programer considers test data for permanent inclusion in the developers data base and approves the program for the next level of testing in accordance with the developers internal procedure.

Developmental Center Test Level 11 (DCT 11) - A test of the individual programs which have completed DCT 1, grouped in their appropriate jobs/cycles. Primary objectives of the DCT 11 include:

- Identify and correct program interface errors.

- Exercise all variations of job flow affected by change to demonstrate technical correctness of the job.

- Exercise all functions affected by change to demonstrate functional correctness of the job.

The DCT II is normally a joint responsibility of programers and their supervisors. A Test Director may be appointed if the developer desires. Formal test plans are not required, but detailed checklist outlining performance standards and I/O requirements are strongly recommended. The developer maintained data base will be supplemented as necessary to meet test objectives. At the conclusion of DCT II the applicable programer will insure that actual output agrees with predetermined results and that the program interfaced properly within the job.

Developmental Center Test Level III (DCT III) - A systems test which exercises multiple jobs interfacing in the total system. The objectives of the DCT III are:

- Demonstrate to the proponent agency and the developer that all change requirements have been met.

- Demonstrate that all cycles of the system affected by change will execute and are technically adequate.

- Demonstrate that all outputs affected by change are correct and functionally adequate.

The DCT III is conducted by the developer, may be monitored by QAD, and is validated by the proponent agency. A formal test plan is required. The developers test data base will be examined by the proponent agency and supplemented as required to thoroughly test changes to the system. At the conclusion of the test, a formal report is prepared and a signed acceptance is required prior to release to the next stage of testing.

Environmental Test - A third-party, technical test of SCP in the form they are to be shipped to customers. The test evaluates machine and human interface by using the computer operations manual, target hardware, representative operator skills, and live data bases captured from field installations. The primary objectives of the ENT are to:

- Insure all jobs affected by change will execute to end of job.

- Successfully load program and JCL changes and execute other implementing jobs necessary to install the systems change.

- Evaluate the accuracy and completeness of the computer operations manual.

The Quality Assurance Directorate is responsible for developing the ENT plan and appointing the ENT director. Upon completion of ENT a formal recommendation of release to the next stage of testing is signed by the Test Director. The ENT is covered in more detail in subsequent discussion.

Field Validation Test - A test conducted at an operational unit with representatives of the user community, proponent, developer, and quality assurance on site. The primary objectives of the FVT are to:

- Demonstrate to participants that the software change satisfies performance requirements of the change.

- Evaluate functional and ADP documentation for accuracy and ease of understanding.

Installation of the SCP and subsequent execution of the cycles/jobs are to be under normal production environment and are the responsibility of the host unit. Therefore, the FVT is the only test conducted in a true production environment. At the conclusion of the FVT a formal memorandum of agreement is prepared and the change is approved or rejected for release to all field users. If the system has multiple version (e.g., different operating systems) a supplemental site is chosen to run the change for a specified time prior to release. (There are also emergency change packages and urgent change packages but the time criticality of these changes dictates an abbreviated testing cycle).

Since the primary focus of this paper is on the Quality Assurance Directorate's role in the maintenance testing cycle, a more detailed look at the resources and methodology employed in the ENT is pertinent. In order to maintain strict third-party independence, a significant amount of duplication has necessarily evolved. For example, separate test data bases for each system are procured and maintained by the QA Directorate. Additionally, system unique card decks, documentation, master files, etc. must be obtained and managed to conduct ENT. The magnitude of these requirements, coupled with the personnel resource constraints imposed on the ENT function, dictates that

the individual testers become generalists, i.e., capable of interpreting systems documentation and overseeing the job/cycle execution (much as a system analyst at a data processing installation), but for the most part unfamiliar with the specific functional output of the system. Thus, the ENT is a technical checkout which insures that the various cycles will execute without abnormal termination. (This check-out, of course, is limited by the constraints of the data base utilized). To maintain the integrity of the ENT, the Quality Assurance Directorate must maintain accountability of the systems change package from the beginning of the ENT until final release to all users. This entails a quality assurance representative delivering the package to the field test, remaining on-site with the developer's representative to monitor any necessary "fixes", and finally performing the necessary actions to cause the creation of the shipment tapes and documentation for each user. These steps are required from a quality assurance point of view, but they create considerable overhead beyond that which would be required to only perform an independent test at a given point in the change package life cycle.

3. Perceived Need to Change Procedures

The ENT, as described above, was established as a testing requirement in 1974, almost certainly because of critical problems with change packages after their installation by Army users. The ENT evolved to insure that change packages would install and run without abnormal termination. Although this may seem to be a basic criteria for developer testing, it was evidently not met frequently enough which caused the creation of the ENT function. (There are many explanations for this including variances in developmental and user hardware, time constraints imposed on the developer, inadequate test data bases, and the all encompassing human error). During the seven years since the implementation of the ENT, many theoretical questions concerning its desirability and effectiveness have been raised. Two of the most pertinent issues are: (1) The value of an independent test in general and (2) The Quality Assurance role within the overall life-cycle of a maintenance change.

The question of the value of an independent test is more complex than it might appear at first glance. Should an independent test be conducted at all? If so, should there be a myriad of independent test activities (or individuals) associated with each system or one central test activity? At what stage in the developmental cycle should the independent test activity erter the process; that is, should they insist on examining the change package at the end of the "assembly line" much as a quality assurance worker in a factory? Or, would it be better to test early in the programing process thereby detecting fatal flaws, but negating the effect of a final independent test before final release? There is considerable academic material available on the subject. One theory on the desirability of a test similar to ENT is that this type test is one of two extremes. At one end of the spectrum (the ENT is at this end) is the person who treats the software as a "black box" and attempts to provide the majority of possible inputs, verifying that the programs meet the specifications. At the opposite end is the individual who studies the program logic and, for example, tries to design test data for each potential branch in the logic. Both methods are held to be too extreme and, in reality, impossible to fully accomplish [2]. But this theory deals with the broad arena of software development and testing. USACSC, specifically QAD, in recognition of the fact that perhaps the "text-book" solutions were not being utilized, commissioned several studies of the quality assurance program. One of these studies concluded that the Environmental Test should be combined with the DCT III to be conducted as the latter is now. This report suggests that "independent third party testing is only a good idea if it is an oversight activity. In gaining the independence of a third party test, so much expertise is lost from the proponent and development groups that the results are bias free but too shallow to be worthwhile. Only by keeping those who know the most about the software in the testing loop can their deep knowledge be used to speed up the cycle of error detection and correction. While there is always danger of bias where the test group has special knowledge, this danger is less than the one of trivializing the tests. Whenever objective measures of test quality are available, the best of both worlds can be obtained: The expert group can conduct the test, and the independent group can verify (in the sense of an auditor) the objective measure is satisfied [3]." A second study, performed internally, reached similar conclusions in noting that "The ENT is excessively expensive and time consuming for the purpose it serves.... This is not a comment on the diligence or dedication of

the personnel who conduct ENT; it is simply that the ENT was devised as a measure to cure a symptom, that symptom being fielding packages which did not load or execute.... To do this is essential; however, it can be accomplished easily and cheaply by conducting all or part of the DCT Level III on a broadcast ready package and target hardware [4]."

Thus both general literature and specific studies appeared to point toward the same conclusions. But it would be insufficient to separate independent testing as an entity unrelated to its role in the context of the complete quality assurance program; i.e., if the ENT is not paying sufficient dividends, what will? Again there is abounding literature concerning an ideal QA program, some general in nature, some dealing specifically with USACSC. One of these general works views quality assurance as only one of three software assurance methods. Figure 5 graphically depicts these three methods. The interesting point is that the quality assurance portion of this triad is responsible for conducting design reviews, audits, walk-throughs, and witnessing acceptance testing [5]. Returning to the specific studies of QAD, there is a common thread exemplified by the following quote: "No enterprise can afford to use more people to check quality than to create the objects to be checked. The solution is to make QAD an oversight activity [6]." A major conclusion of virtually every report was that QAD should concentrate more of its resources in up-front activities such as simulation, optimization, design reviews, and monitoring the testing conducted by the developer.

4. Revised Testing Procedure

With the stage being so obviously set for a change in the Command's QA program, particularly regarding independent testing, the following changes were instituted on September 1, 1980:

- Selected standard systems would undergo revised testing procedures. Change packages for these systems would not be subjected to ENT, but instead would have a DCT Level III with ENT objectives and be conducted on target hardware. The developer for these systems would be the Command representative at the field test and QAD participation in these final tests would be virtually non-existent.

- On a time-phased basis, additional systems would undergo these revised testing procedures. Eventually, all USACSC systems would transition to the revised testing procedures.

- Quality Assurance resources would gradually be shifted to the upfront activities previously mentioned, i.e., simulation, optimization, design reviews, and early test monitoring.

The impact of these changes can not be underestimated. The quality of products affecting the operation of the entire Army was at issue. The revised procedures were not received with overwhelming acceptance by the various developers in the command, although their objections were most likely keyed by a perceived increase in workload without a corresponding increase in manpower. The theoretical wisdom of the change was not significantly

FACTOR	QUALITY ASSURANCE	ACCEPTANCE TESTING	INDEPENDENT V & V
PRIMARY OBJECTIVE	ENFORCE STANDARDS	DEMONSTRATE ACCEPTABLE PERFORMANCE	ELIMINATE CRITICAL ERRORS
ORGANIZATION	INDEPENDENT GROUP	DEVELOPMENT GROUP	INDEPENDENT GROUP
RELATIVE SIZE	SMALL STAFF, LONG DURATION	LARGE STAFF, MEDIUM DURATION	MEDIUM STAFF, LONG DURATION
APPLICABILITY	DELIVERABLE SOFTWARE	OPERATIONAL SOFTWARE	CRITICAL SOFTWARE
MAJOR STRENGTH	COST-BENEFIT RATIO	WIDELY ESTABLISHED	EXTREMELY EFFECTIVE
MAJOR WEAKNESS	DIFFICULT TO INITIATE	SUBJECT TO TUNNEL VISION	REQUIRES ADDITIONAL RESOURCES

Figure 5. Comparison of Software Assurance Methods

questioned. Nevertheless, there was sufficient apprehension to cause a delay in the change (originally scheduled for July 1, 1980) and a reduction in the number of systems initially affected. Regardless, the mechanism to eventually accept the revision for all STAMMIS was in place.

It was also recognized that any major change in testing of software was palatable only if some method of tracking the effectiveness of the new procedures was devised. In this regard, QAD instituted an after the fact tracking for change package releases called STAMMIS Release Evaluation and Performance Profiling. This methodology, in its early stages, is primarily the gathering of pertinent information about a particular change package release in a format which facilitates the comparison of this data from one release to the next. Significant trends can then be charted and reported to managers. The many elements of information being gathered can be divided into two major catagories: Those which describe or differentitate vis-a-vis other systems (these factors include size, age and % programs changed in a release) and those which provide some indications as to the effectiveness of the recent change packages for a system (including number of incidents submitted by users against the system, number of changes requested against the system, number of emergency and/or urgent changes required and frequency of normal change package releases.) Although the system is in its early stages of development, it is theorized that it will be used as an aid in answering questions related to the effect of change on system stability, the optimal time between change package releases, the relationship of system size to stability, and other pertinent quality indicators.

5. Conclusion.

The Quality Assurance Directorate of the US Army's Computer System Command has undergone some major "soul-searching" particularly with regard to its role in independent testing. As a result of this self-evaluation, the changes detailed in this paper are being gradually implemented. Six systems have been placed under the new procedures and all have experienced one or more releases of systems change packages with the revised testing methodology. No significant problems have been encountered and, for the most part, the developers are optimistic about their increased responsibility and role in the complete testing of their systems. Undoubtedly there will be set backs, perhaps even a decline in the quality of USACSC products for a period of time. In fact, it would be an anomaly if problem areas did not occur. However, in the long run the shifting of QAD resources described herein can result in significant savings in the cost of software maintenance while actually strengthening the QA program within the Command.

References

[1] USACSC Manual 18-1

[2] Meyers, G.J., Software Reliability, John Wiley and Sons, New York 1976

[3] Hamlet, R., Testing of Data Processing Software, contract report for Battelle Columbus Laboratories, 1980.

[4] Fox, M. and White, P. Report of QAD Special Study on Testing, Memorandum ACSC-QA, 1979.

[5] Fujii, M. S., A Comparison of Software Assurance Methods, Logicon, Inc.

[6] Hamlet, R., Testing of Data Processing Software, Contract Report for Battelle Columbus Laboratories, 1980.

HISTORICAL FILES FOR SOFTWARE QUALITY ASSURANCE

Walter R. Pyper, Ph.D.

Tracor, Inc.
2007 West Olive
Fullerton, CA 92633

This paper summarizes a combination of ideas, adapted from software engineering and other disciplines, which provide insight into software life cycle management activities. This paper is written for an audience which may not be familiar with software engineering terminology, and some ideas and terminology are taken from related file maintenance and systems analysis fields. The impact of historical files, which represent the software development and testing activity, upon traditional software configuration management (CM) and software quality assurance (SQA) is examined. The implementation of historical files is explored. Both advantages and disadvantages are indicated. The relationship to keyword cross-referencing is discussed.

Keywords: Configuration management (CM); cross reference; data structure; design, documentation; functional; historical file; keyword; life cycle management; requirements; software quality assurance (SQA); specification.

1. Introduction

The intent of this paper is to summarize a combination of ideas, adapted from software engineering and other disciplines, which provide insight into software life cycle management activities. This paper is purposely written for an audience which may not be familiar with software engineering terminology, so some ideas and terminology are taken from related file maintenance and systems analysis fields.

Software configuration management is an area of software quality assurance which is concerned with controlling the current version of each document, program listing, and other items associated with a software system. In the maintenance of such a configuration management system, some key decisions must be made regarding which information to keep and which to throw away. Ideally, nothing is discarded from program inception to system delivery and on into system maintenance. The challenge to the configuration manager is to develop a reasonable update procedure for documents, program listings, tapes, etc., so that there is traceability from one version to the next and so that the current state of documentation matches the supported software. Some trade-offs are necessary. In some cases, previous versions of documents, listings, tapes, etc., cannot be physically maintained; and even if resources were provided to do that, the cross-reference system between current and all other versions

of these items requires an additional resource which may grow to eclipse the originally envisioned software system development activities.

Although it is generally difficult to decide what to throw away when a new version of a controlled item is produced, one solution has been to keep copies of the previous two versions. Of course, one major change followed by three successive minor changes eliminates the version reflecting the major change from the configuration management system, so any such arbitrary rule may cause disaster in terms of traceability. See Figure 1.

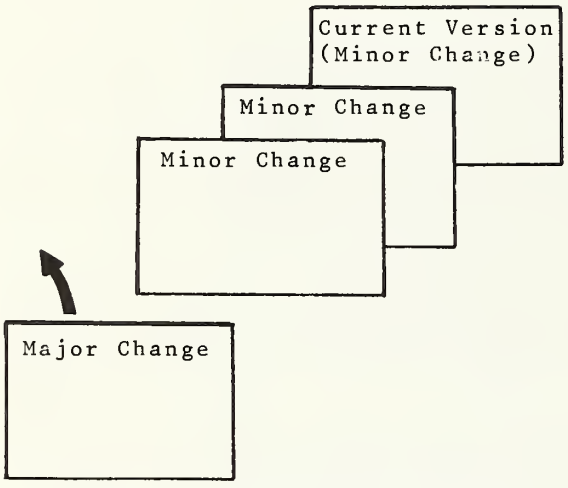


Figure 1. CM Update Procedure

The idea of maintaining a historical file which cross references all updates in each controlled item is proposed as a way in which a software configuration might be controlled appropriately. The idea is a fairly straight-forward one, but its implementation does require discipline which may not traditionally be found in the configuration management arena. The application of historical files to wider areas of software quality assurance is pursued in Section 3, entitled "Wider Application to Software Quality Assurance (SQA)".

2. Applications to Configuration Management (CM)

The basic procedures for software CM employ tools and techniques which can produce the current version of a controlled item (document, program listing, etc.) and indicate the relation between any two items. An equally important CM capability is to produce technical and management information on any item, such as the number of internal program segments in a software task or the location(s) of a given data segment within the entire software system. It is not intended that the use of a historical file might eliminate the need for such traditional CM tools. The major benefit of a historical file to CM is in minimizing the concern over lost information due to elimination of early versions of a controlled item.

The historical file records any change to a controlled item and cross-references that change to all documents, computer codes, etc., which are functionally affected by the change. One challenge is to identify all of these functional relationships and record the cross-references appropriately. Another challenge is to set up the historical file organization in the most useable form at the "beginning". A third challenge is to identify the "beginning". When does one begin to keep a historical file?

Let's start by considering the last challenge first. A historical file may be initiated at any point in the software engineering life cycle, but there is a point of diminishing returns, and there may be a point of insignificant return for the effort invested. If it is possible to begin the historical file near the inception of the life cycle (wherever that is judged to be), there are rewards for keeping track of approved and eliminated ideas, together with the corresponding judgment criteria, during the early stages of requirements generation.

If the historical file is implemented when the preliminary software design document is generated, that document provides an outline for the historical file organization which is close to the ideal, since it reflects the organization of the product itself - the computer code. Valuable information on software requirement specification decisions will have been lost by waiting to initiate the historical file until

the preliminary design document is produced, and it is possible to reorganize the historical file based upon the structure of any document or listing. By automating the historical file, the reorganization procedure may be accomplished less painfully.

The second challenge, that of setting up the historical file organization in the most useable form, has been partially discussed already. Should its organization be arbitrary, should it follow a particular document, or should it be a reflection of the program listing itself? The latter organization, that of the program listing itself, is proposed as the most useful, since the historical file is then keyed directly to the code for design, development, maintenance and other purposes. It is suggested at this point that one of the major sources of software maintenance cost is the cost associated with familiarizing personnel with the software design, preliminary to maintenance updates to the code. Since all documents are written in support of the computer code itself, it seems reasonable that the historical file should be organized parallel to the code.

The content of the historical file has not been discussed to this point, and it is high time we looked at that in some detail. The historical file represents design decisions. What was done? Why was it done? When was it approved? What effect did it have? The historical file should contain, at a minimum, the following:

- Location of Change - supporting information is listed under appropriate historical file section.
- Chronology - date change was implemented in CM controlled item.
- Description of Change - Standardized format of changes to documents, listings, etc.
- Criteria for Change - Standardized comments for reasons behind change.
- Cross-References - What was functionally affected by this change? It is proposed that this cross-reference include usage of a keyword file where keywords and keyword phrases are associated with the historical file section in which the change is being described. These keywords provide a secondary check on code and data segments which might be affected by this change. A second type of explicit cross-referencing which lists all locations in associated documents where changes are required is a traditional part of CM control, and would not necessarily appear within the historical file.

The last element of a historical file entry responds to the first challenge listed earlier. The ability to adequately cross-reference the changes to a software system and provide checks against inadvertent "ripple effect" of a change has plagued software designers and maintenance personnel in the past. The idea suggested here is to associate keywords and keyword phrases with each program segment (subroutine, procedure, etc.) of the computer code and consequently with the associated historical file section. These keywords and phrases provide functional relationships between code segments which yield a semi-automated control by listing all code and data segments which share the keywords and phrases contained in that historical file section. See Figure 2 for a Fortran example.

```
      SUBROUTINE EXAMPLE
C-----3.4.1
C-----PURPOSE
C     .
C     .
C     .
C-----KEYWORDS
C     .
C     .
C     .
```

Figure 2. Keyword Section in Preamble

Admittedly, a lengthy check-list of code and data segments might result from the cross-reference procedure using keywords as the means for relating an altered historical file entry to the balance of the entries. This procedure is amenable to

refinement by means of sub-categories, so that the check-list is not so extensive. The intent is to be exhaustive in the cross-reference exercise, so it may be well to employ manual sub-category sorting techniques rather than automate to the degree that the intrinsic value of such a cross-referencing procedure is lost.

In what respect have we aided the CM process? If the historical file were not available we might expect that the following deficiencies would exist in our CM system:

- The history of changes to a given code segment or document section might be difficult to trace back more than a few versions prior to the current version.
- The reasons for changes to design, requirements, etc., might be obscured or non-existent with traditional CM techniques.
- The cross-reference between sections of documents or segments of code might not exist in a useable form for maintenance or design changes, thus amplifying the aftershock of CM activity.

The additional savings associated with a reduction in the need for maintenance of multiple prior versions of listings and documents may be the strongest reason for implementing a historical file.

3. Wider Applications to Software Quality Assurance (SQA)

Upon test or examination of any code segment by means of SQA tools, the results of such examination should be recorded in an easily accessible central file. The ramifications of such testing and the ability to duplicate the test should be referenced within that same central file. The intent of these suggestions is to minimize maintenance cost - the greatest potential cost of software if redesign and major updates to the software are included in maintenance costs.

It is suggested that the historical file, with an organization which reflects the code structure, is the appropriate place to record results of the application of SQA tools and testing activity. In order to formalize the organization of the historical file and tie it to the delivered code structure, it is recommended that both the code and the historical file be structured with a numbering scheme which is common to the final design document. In particular, the individual sections of the final design document should be implemented within the code in single program segments (subroutines, procedures, etc.) or in groups of program segments (e.g. tasks). The numbering system associated with those individual sections of the design document may be included with the preamble or header comments in the individual program segments and may be likewise repeated in the historical file. In the event that a program segment does not match a design document section uniquely, as in the case with certain utility programs, an appropriate subsequence numbering is possible so that the correlation is maintained.

The advantages to the maintenance of such a common numbering scheme extend beyond configuration management activity, discussed earlier. All parent and descendant documents which support the code may be referenced directly to code segments. Test results at all levels may be referenced directly to code segments. A historical file is easily organized and maintained based upon such a numbering procedure.

The effort involved in maintaining the common numbering system is insignificant compared with the potential advantages gained. The historical file may be organized in such a fashion that proposed updates to any code segment are complemented by extensive background on prior testing, validation and verification associated with that code segment. Any data segment which is affected by a proposed change should be similarly categorized in the historical file.

If the supporting data base is organized by such a structured procedure the possibility of unexpected errors due to code or documentation update can be significantly diminished.

There is no substitute for manual review and correlation of

some information, but the presence of a historical file with built-in features which reduce the search and correlation process to a standardized procedure is a significant aid.

4. Implementation of the Historical File

The individual entries within a historical file may be formatted under the appropriate section number to include the items indicated in Section 2, plus any others that are identified by the user. Each entry may be cross-referenced in some fashion, and the section in which the entry resides is cross-referenced by keyword as well as by section number which matches the final design document section number.

The historical file resides on an edit-driven system with the ability to expand, concatenate and delete file sections, as well as to search and sort on given keywords or other attributes. The ability to rapidly access any portion of the file by numerical pointer is a key to its usefulness. The relationship between keywords and numbered sections is maintained within the preamble or header of program segments (subroutines, procedures, etc.) and duplicated in the historical file unless it is critical to save space in the historical file and not repeat the keywords within the historical file.

Advantages to implementing a historical file have been outlined in Sections 2 and 3. They are summarized here:

- Preserve design history of software.
- Provide cross-reference to control structure and data structure changes.
- Provide essentially unlimited backup to current version of all items under CM control.
- Provide insight to design processes for new implementors.
- Provide detailed history to support maintenance activity.
- Provide management tool for extraction of significant software update activity.
- Provide central repository of all testing and SQA activity.

Resistance to the implementation of a historical file is generally prompted by lack of resources, to wit: time, money and personnel. The return on investment for such a tool is generally not realized to a significant degree until all major design work has been completed; although structuring the development activity so that design changes at all levels are carefully recorded and organized yields some very important benefits.

Each individual involved in the development of the software may go to the historical file to gain understanding of otherwise tersely-written documents. A significant time savings is effected, and added insight into total system requirements is derived by the reviewer. Attempts to "second-guess" the system architects are diminished, and due to the increased level of information available, the implementor is given greater confidence in the basic design. If that is not the case, useful suggestions for improvement will be forthcoming at an early stage where the cost of change is small by comparison to later stages.

The details of implementation of a historical file are, to a great degree, dependent upon the resources and particular needs of the sponsoring organization. Some of the advantages and disadvantages of starting the historical file before the final design document is produced have already been discussed. The suggestion here has been to start as soon as possible and structure the historical file so that it may be altered at a later date to reflect the final design document. This is no easier task than restructuring the requirements document in terms of the final design document, so this author recognizes that the task of restructuring is probably not amenable to automation for many complex projects.

The process of correlating keywords within the requirements document and the initial design specification allows a semi-automated procedure for relating the preliminary historical file (based upon the

requirements specification) to the final historical file, thus anticipating the organization of the final design specification. See Figure 3.

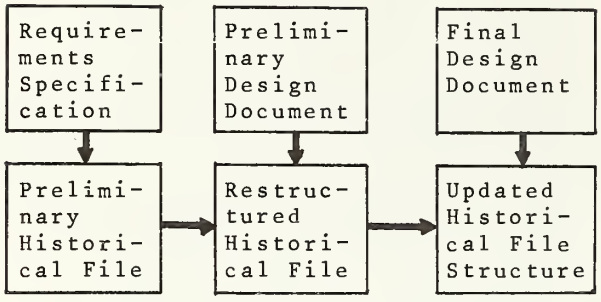


Figure 3. Historical File Update Process

If the reader feels that we are beginning to magnify the software development effort beyond reason, consider the effect of doing the cross-reference exercise to generate a representative historical file. The mechanism of thinking about cross-references in detail provides the designer with insight perhaps not otherwise accessible, and the added attention to that detail should reveal problems with consistency and completeness in the design. This activity occurs at an early enough stage to make changes which might be impossible once the coding cycles have begun.

5. Summary

The concept of a historical file has been utilized effectively in many areas of systems and information science. It is a standard tool in the control of organizations and large projects. It has been somewhat neglected in software engineering, possibly due to the apparent complexity and cost of maintaining a separate historical file when so much was already being recorded and controlled by CM techniques.

One key element which has been lost is the history of design decisions which provide great insight into sometimes obscure reasons for the existence of data and control structures in their final, delivered form, or after several maintenance versions have been generated.

By organizing such a file to match the structure of the delivered code, and providing the ability to cross-reference related documents, test results, SQA results, and other information, cost effective CM and software maintenance is more probable. If data structure is also considered in the generation of the historical file, heretofore unexploited software maintenance techniques are indicated.

The intent of this paper has been to generate interest in the use of historical files for reducing software maintenance costs and to provide greater visibility into the software development process, thus reducing the cost of design change by moving more of the detailed review of design decisions toward the beginning of the software development process.

The keyword cross-reference approach has been suggested as logical complement to the historical file. This provides a second check on maintenance activity, and yields likely nooks and crannies to search for "ripple effect" errors. The use of keywords and phrases is explored in more detail in [1].

References

[1] Pyper, Walter R., The Effective Use of Comments, Tracor, Inc., 20 March 1981.

CPEUG81

ADP Cost Accounting and Chargeback

A STEP-BY-STEP APPROACH FOR DEVELOPING AND IMPLEMENTING A DP CHARGING SYSTEM

Kenneth W. Giese
Dean Halstead
Thomas F. Wyrick

Directorate of System Evaluation*
Federal Computer Performance Evaluation
and Simulation Center (FEDSIM)
Washington, DC 20330

Charging for data processing (DP) services refers to distributing the costs of providing DP services to the users who receive the services. The distribution of costs requires definition of the basic DP services, the resources used to provide the services, and the costs incurred to obtain and make use of the resources. A charging system is comprised of two subsystems: the rate-setting and the billing subsystems. The rate-setting subsystem incorporates procedures for forecasting the use and the total cost of each service and establishing the rate to be charged for each unit of service. The billing subsystem includes procedures for monitoring the use of services, applying the rate for each service unit to compute the total charge for the services each user receives, and reporting the charges to the user and pertinent accounting groups. Organizations may or may not collect funds to recover the costs of DP services.

The Federal Government has established policies that call for distributing the "full costs of operating DP facilities to users according to the service they receive." The National Bureau of Standards (NBS), assisted by the Federal Computer Performance Evaluation and Simulation Center (FEDSIM), is developing a set of Guidelines to assist Federal DP management in implementing the charging aspects of this policy. This paper describes a step-by-step methodology for the development and implementation of a charging system that will form the basis of the Guidelines. The material presented in this paper represents an overview of the preliminary results of the Guidelines development project and may change during the review of the draft Guidelines document. Selected technical and policy references are cited to encourage further investigation.

Key words: DP cost accounting; DP cost allocation; chargeback; pricing; standard costing; chargeout; charging; charging system.

*The views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinions or policies of FEDSIM or of any other person(s) or agency associated with the Government. Moreover, this paper contains the authors' summaries of official documents and, therefore, may not reflect the full intent of those documents.

1. Introduction

On September 16, 1980, the Office of Management and Budget (OMB) issued Circular No. A-121, "Cost Accounting, Cost Recovery, and Inter-Agency Sharing of Data Processing Facilities." This Circular requires Federal agencies to implement policies and procedures to (1) account for the full cost of operating data processing (DP) facilities; (2) allocate all DP costs to users according to the DP services received; (3) evaluate inter-agency sharing as a means of supporting major new DP applications; (4) share excess DP capacity with other agencies; and (5) recover the cost of inter-agency DP sharing. The Circular specifies that agency procedures for cost accounting and charging must be consistent with the guidance provided in Federal Government Accounting Pamphlet Number 4 (FGAP 4), "Guidelines for Accounting for Automatic Data Processing Costs," U.S. General Accounting Office (GAO), 1978.

While Circular No. A-121 and FGAP 4 describe general criteria that agencies should use for cost accounting, Federal DP managers need technical guidance in the area of charging. The National Bureau of Standards (NBS) is developing a Federal Information Processing Standard (FIPS) Guidelines to provide the needed guidance. The Guidelines will incorporate the "best practices" as described in the literature and as used by exemplary managers in the Federal Government and private sector.

This paper presents an introduction to charging for DP services and summarizes a step-by-step methodology for developing and implementing a charging system. The material described in this paper represents an overview of the preliminary results of the Guidelines development project. These results may change during further review of the draft Guidelines. Selected technical and policy references are cited to encourage further investigation.

2. Introduction to Charging for DP Services

The term charging for DP services refers to the process of distributing the costs of providing DP services to those who receive the services. The procedures used to charge and recover costs for services will vary from DP facility to DP facility. Services, resources, and their interrelationships are the fundamental concepts that must be understood in order to understand how to charge. The term "service" refers to the work performed by the DP facility for its users. In order for work performed by the DP facility to be classified as a service, the work must be measured by only one metric and must have a user billing rate (price) associated with it. The metric is referred to as a service unit and is the smallest category of work for which the users can be charged. Services can be as simple as computer processing, with CPU seconds as the service unit, or as complex as a machine resource unit (MRU), which is an algorithm that combines several different services and that uses the number of MRU's as the service unit.

The term "resource" refers to the items employed by the DP facility to provide one or more services. In order for a resource to be included in the charging system, the DP facility must incur a cost for obtaining or using the resource. Examples of resource categories that are used in DP facilities are those listed in A-121 and FGAP 4: personnel, equipment, software, supplies, contracted services, space occupancy, intra-agency services and overhead, and inter-agency services. The total cost of each resource is used as the basis for determining the cost of each service, which is in turn used as the basis for calculating the billing rate that will be charged for that service. The concepts of accumulating and accounting for the total costs of DP services are described in detail in FGAP 4.

The billing rates for the DP facility's services typically are developed by dividing the total cost of a service by either the projected utilization or available capacity for the service, such as the number of hours the service is available for use. Usage accounting procedures collect and store the number of service units for each service utilized by each user. These data are reduced and reported to the respective user. If the DP facility is recovering its costs, then the billing rates for the service units are applied, the charge for the work is calculated, and a bill is prepared and sent to the user.

The procedures used to charge for services are referred to in the draft Guidelines as a "charging system." Developing a charging system requires many decisions that can affect not only the DP facility, but also the entire organization. Examples of the decisions are (1) whether to charge the users for all of the resources used to provide a service, or to charge for only certain resources; (2) whether or not to use the charging system

to try to influence service usage patterns; (3) which services the DP facility should charge for; and (4) which resources are used to provide a particular service. These types of decisions are the main focus of the draft Guidelines.

3. Functional Description of a Charging System

An operational charging system is composed of two subsystems: rate-setting and billing. (See Figure 1.) The rate-setting subsystem supports the development of the rates charged for each service. The billing subsystem records usage and applies the rates to compute the total charges for the services each user receives.

3.1 Rate-Setting Subsystem

The rate-setting subsystem of a charging system is most often linked to the organization's fiscal cycle, which subsequently dictates the schedule of the subsystem's operation. The frequency of a typical cycle is yearly. The main objective of the rate-setting subsystem is the setting of billing rates. Billing rates are the prices users are charged for the services they receive. Most organizations attempt to keep billing rates constant throughout the fiscal cycle to prevent disruption of budgets, which explains the interdependence between the subsystem and the organization's fiscal cycle. The work performed in the rate-setting subsystem is best described by dividing it into four procedures: usage forecasting, cost forecasting, setting billing rates, and DP management accounting. Extensive interaction occurs between the various procedures, and the work is not always completed in the exact order shown in Figure 1. Each procedure is briefly discussed below.

The first procedure of the rate-setting subsystem is usage forecasting. Usage can be forecast by one or more of several techniques. Some examples of forecasting techniques are surveying users to estimate the amount of service they will require, performing a trend analysis based on historical usage data, or a combination of the two. After the forecasts have been compiled and validated for accuracy, the DP facility determines if available DP capacity can support the forecasted usage. If not, steps must be taken to adjust either DP capacity or the number of service units that users are requesting. Additionally, the DP facility evaluates the usage forecasts for each service to determine if the DP facility should continue to offer the service.

Cost forecasting involves estimating the costs incurred to provide the resources needed to support each service. The objective of cost forecasting is to estimate all costs for the fiscal cycle. There are several techniques often used for cost forecasting. One of the more common techniques involves extrapolating the DP facility's current operating budget into the planning period by itemizing changes to the budget reflecting estimated future costs. Once cost forecasting is completed, the costs and benefits of each DP service are evaluated by comparing the forecasted usage and cost estimates to determine if that service can be economically provided. Cost forecasts can be used as a basis for the DP facility's budget requests. Together, usage and cost forecasting provide the data needed to set billing rates.

Setting billing rates involves allocating the forecasted DP costs to the various DP services. This allocation involves determining the portions of each resource or group of resources (resource center) used to support each service. The total cost of each service is divided by either the forecasted capacity or the forecasted usage for that service; the result is referred to as a "standard rate." The standard rate is the minimum amount that must be charged for each service unit to fully recover the cost of providing that unit. Differential surcharges or discounts may be applied to each standard rate, if appropriate, to develop the final billing rate for each service. The standard rates may also be used to develop charging algorithms, if the DP facility is using them. Charging algorithms are the equations used to compile the usage of several different services into a total usage unit, such as the MRU described earlier. The billing rate for the total usage unit is calculated using some combination of the standard rates of each of the services in the algorithm. The billing rates are then published and used as a basis for developing the users' DP budget requests.

The final procedure in the rate-setting subsystem is a set of activities referred to as DP management accounting. These activities include budget planning and the formulation of budget requests; establishing and maintaining accounts and accounting information for users and the DP facility; and providing any interface that may be

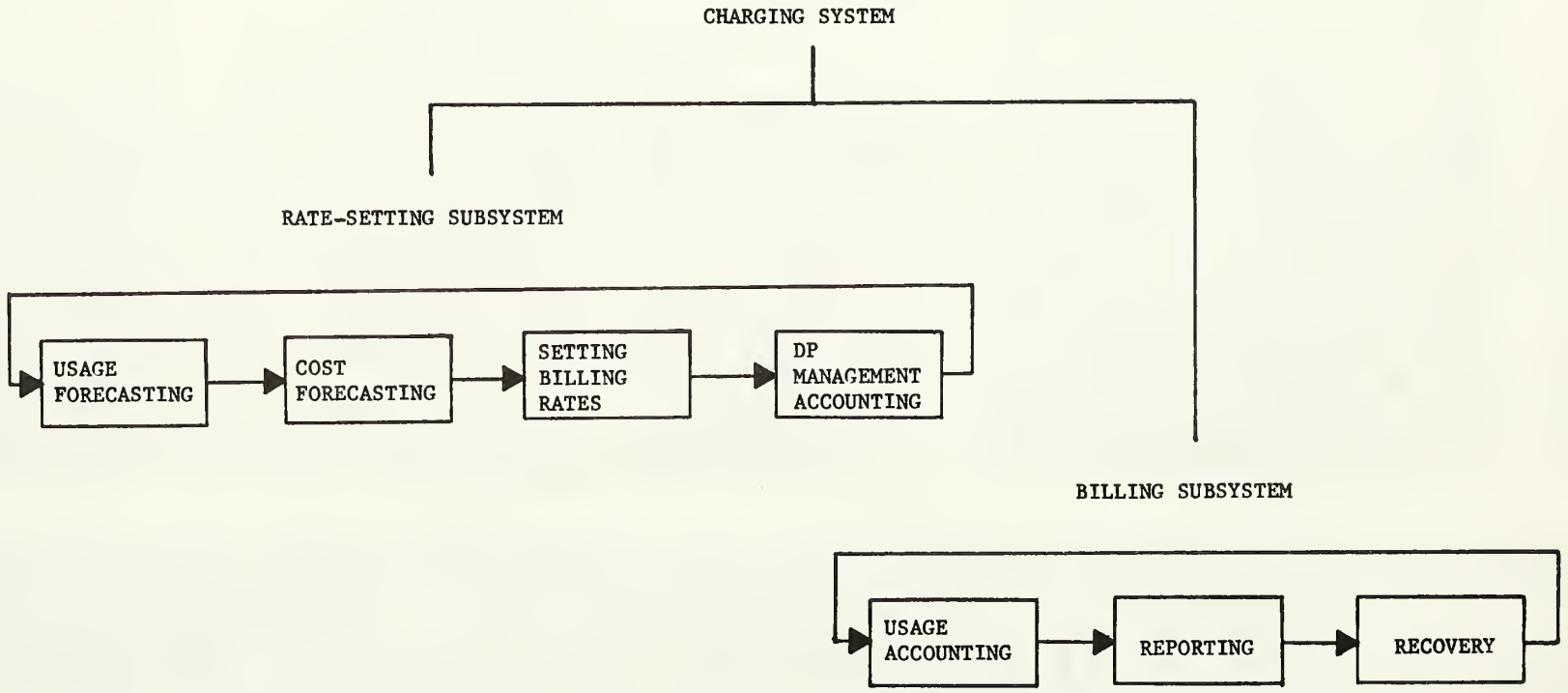


Figure 1. Components of a Charging System

required between the DP, budgeting, and accounting groups of the organization.

3.2 Billing Subsystem

The billing subsystem consists of the procedures employed to monitor the usage of the services; to prepare and distribute reports of each user's service utilization and charges for that utilization; and to recover costs. If the DP facility is recovering costs, usage reports may then be viewed as bills. Service usage is typically monitored continually; reporting typically occurs monthly; and cost recovery, which may or may not occur, typically follows a schedule that is closely tied to the fiscal cycle of the agency.

Usage accounting procedures, in the context of charging, refer to recording the service units received by each user. Usage of DP equipment and software is typically monitored by job accounting software run in conjunction with the computer's operating system. Manual procedures, including time sheets, data entry logs, tape mount logs, and other manual logs may be used for personnel and other resources. Data from the job accounting software and the manual logs are compiled to produce records of total services received by each user during a given period.

Reporting consists of the procedures employed to reduce the usage accounting data, to apply the billing rates to the results of the reduction, and to produce reports summarizing the total service utilization and charges for that utilization for each user. Many DP facilities actually provide their users with a report at the end of each computer session or run which summarizes the services received for that session or run and the estimated charges for those services. If the DP facility has chosen to recover the charges for its services, then in most instances the reports that have been produced can be viewed as the bills used for recovery.

Recovery consists of the procedures employed for the collection of funds. There are several different techniques that can be used to handle the recovery of charges. One technique is to have the user deposit an amount of money into an account maintained by the DP facility. The DP facility then debits the user's account every time the user utilizes the DP facility's services. In this situation, the reports or bills reflect the charges that have already been removed from the user's account. Another technique is to have the user send funds to the DP facility for each bill received. Whatever technique is used by a DP facility, it must fit in with the policies and procedures that each agency has for the transferral of external and internal funds. The agency's accounting department typically handles all of the agency-level record-keeping for, and actual movement of, the funds.

4. Outline of the Charging System Development and Implementation Process

The draft Guidelines discusses in detail four phases and eleven steps for developing and implementing a charging system. (See Figure 2.) This section provides a summary of the major tasks required for each of the eleven steps.

4.1 Planning Phase

The planning phase incorporates two steps: (1) determining the management structure and approach to developing and implementing the charging system; and (2) setting the objectives for the DP facility, for charging for DP services, for the charging system, and for the charging system development project. These steps result in the preparation of a high-level management plan for the development project.

"Step 1" Determine the Management Structure and Approach

There are four major tasks in Step 1. The tasks are primarily the responsibility of the organization's management. First, the level of management involvement and support in the design, development, implementation, operation, and maintenance of the charging system should be determined. This involves defining the degree of organizational commitment to charging for DP services, the charging system, and the charging system development project. Second, a charging team should be formed. The team should consist of at least one representative from each functional group in the agency that will be affected by the charging system, including DP, management, budgeting, accounting, and the users. Third, the charging team should identify the major types of work that will need to be performed and assign the responsibility for completing the work to the appropriate team member or the group they represent. Fourth, the methodology that the organiza-

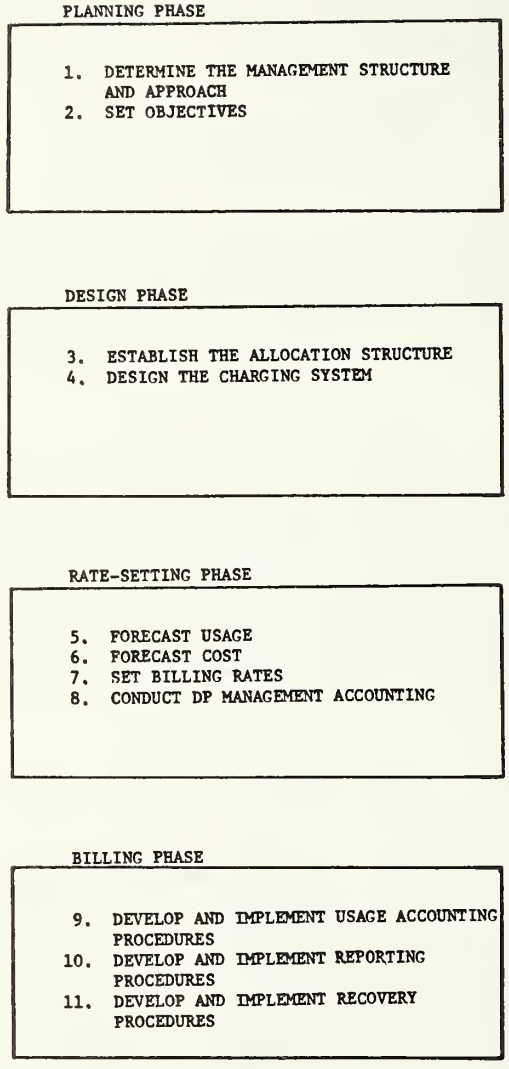


Figure 2. Charging System Development Phases and Steps

tion will use in the design, development, implementation, operation, and maintenance of the charging system should be defined by the charging team. The team's decisions should be documented in a formal Project Plan which outlines the individual tasks to be performed in each subsequent phase and step.

"Step 2" Set Objectives

Step 2 consists of four major tasks. These tasks are coordinated by the charging team, whose objective for this step is to provide a formal definition of the objectives for the DP facility, for charging for DP services, for the charging system, and for the charging system development project. First, the management-level DP objectives should be defined to indicate how DP relates to the organization as a whole. This involves determining the relationship of the DP facility to the organization, the level of maturity of the DP facility, and whether approval for DP usage should be via the DP facility's or user's budget. Since these objectives affect the charging system and the entire organization, they should be coordinated with upper management and all other involved groups. Second, the DP facility's or agency's objectives for charging for DP services should be determined. Examples of these objectives are to (1) increase the accountability of DP to upper management and the users, (2) improve the relationship between charging and the capacity planning process, (3) recover the cost of operating the DP facility, (4) encourage more efficient use of resources by the users, (5) encourage the DP facility to keep prices competitive with outside vendors, and (6) influence the behavior of users regarding their use of the resources.

Third, the objectives of the charging system should be determined. These objectives should define how the charging system should be structured and designed. There are ten primary objectives of a charging system identified in the draft Guidelines document: (1) repeatability, (2) understandability, (3) equitability, (4) auditability, (5) adaptability, (6) inexpensive to operate, (7) easy to implement and maintain, (8) controllability, (9) stability, and (10) simplicity. Many of these objectives are not compatible and the charging team must work to explore trade-offs between conflicting objectives.

Fourth, the objectives of the charging system project must be defined. This involves setting the goals, schedules, and

budgets for accomplishing the phases, steps, and tasks defined in the Project Plan.

4.2 Design Phase

During the design phase, the objectives set forth in the planning phase are used to direct the conceptual development and preliminary design of the charging system. The objectives of this phase are to establish an allocation structure, which is a matrix to be used to assign the resources to the services, and a global charging system design. During this phase, the global requirements for the charging system are identified and alternatives for satisfying the requirements explored. The allocation structure and charging system design provide a starting point for the detailed design, development, implementation, and operation of the rate-setting and billing subsystems. The global charging system design serves to coordinate the individual steps and tasks of subsequent phases. The design phase is coordinated and performed by the charging team members.

"Step 3" Establish the Allocation Structure

Step 3 consists of seven major tasks. First, the methods for allocating the costs of the resources to the services should be determined. There are two decisions which must be considered, (1) should the costs of all or only a subset of the resources be allocated to the services, and (2) should billing rates be based on standard or actual costs. The first decision is self-explanatory and is a function of the policy of the DP facility or the agency in which it resides. The second decision concerns how often the billing rates are set. Billing rates based on standard costs are set once for the fiscal cycle; billing rates based on actual costs are changed as often as necessary to ensure that they reflect the actual costs to produce them. Second, the charging methods or bases should be determined. There are two charging method decisions that should be determined: (1) should the rates be based upon projected usage or estimated total capacity; and (2) should the services offered by the DP facility be charged using a single factor, multiple factors, or an extension of the multiple factors method? (These are discussed in detail in the draft Guidelines.)

Third, the services that the DP facility wants to provide should be itemized. The service unit for each service should also be defined. Fourth, the resources that the DP facility uses to provide each service should be itemized. Fifth, the centers, or logical groupings used to accumulate the resources and services, should be identified. Sixth, the structure for allocating the costs of the resources and resource centers to the services should be constructed. The actual costs of the resources are not placed into the allocation matrix at this time, because they are not yet known. Seventh, the charging algorithms, if they are to be used, should be developed.

"Step 4" Design the Charging System

Step 4 consists of four major tasks. First, the global functional requirements of the system should be identified, analyzed, and documented. The requirements for the charging system that are of primary concern are the basic inputs, processes, and outputs needed for the various procedures. The allocation structure provides a basis for defining requirements for collecting, processing, and reporting information about resources, costs, and services. Second, the global functional requirements should be used to define and document the data requirements of the system. The amount and types of needed and available usage and cost data should be closely examined and defined. The data requirements document is very important and should be continually refined throughout the charging system development project. All design decisions should be checked against the availability of the data needed to develop and implement the procedures and systems that those decisions dictate. Third, the basic alternatives for satisfying the functional and data requirements of the charging system must be explored. Examples of the decision trade-offs for each of the procedures of the charging system are manual versus automated, in-house versus purchased, and existing versus new. The potential costs of each alternative for each procedure should be weighed against its benefits. These decisions are iterative and may result in the need to modify the objectives and requirements of the system. Fourth, based on decisions made in the first three tasks, a global charging system design document should be compiled, reviewed, and eventually approved or disapproved by the charging team. Approval leads to the next two phases, during which the detailed designs are produced, the system and subsystems are developed, and finally they are implemented and operated. Disapproval means that

earlier phases, steps, and tasks must be re-evaluated and, if necessary, repeated until approval is obtained by the involved parties.

4.3 Rate-Setting Phase

This section describes the four steps to follow in setting rates. The agency's standard DP systems development techniques should be used in conjunction with this section to structure the detailed design, development, implementation, and operation of the rate-setting subsystem. During the rate-setting phase, the detailed decisions concerning the type and structure of the rate-setting subsystem are made. This section presents information which is pertinent to the operational rate-setting subsystem to aid in its detailed design, development, implementation, and operation. The functional procedures include the forecasting of usage and costs, setting of billing rates, and performing DP management accounting tasks.

"Step 5" Forecast Usage

Step 5 consists of four major tasks. First, users' projected workload requirements should be obtained. The projections should be in terms of the service units that are established in the allocation structure. Second, the forecasts should be analyzed and used to determine the capacity needed by the DP facility to provide those forecasts. Third, discrepancies between available or obtainable DP capacity and usage forecasts should be resolved. Fourth, the allocation structure should be reevaluated and, if necessary, restructured to incorporate the resolutions between the users' forecasts and DP capacity.

"Step 6" Forecast Costs

Step 6 consists of four major tasks. First, an estimate of the budget that will be proposed for the DP facility, for the planning period of the charging system, should be developed or obtained. Second, additional cost data that are either not contained in the budget estimate or not contained in the detail that is needed should be obtained. Third, the cost data should be analyzed and the costs for the resources and resource centers should be calculated. Fourth, the allocation structure should be reevaluated. Resources and resource centers for which no cost could be obtained should be removed from the allocation structure or consolidated under other categories.

"Step 7" Set Billing Rates

Step 7 consists of five major tasks. First, the estimated costs of the resources and resource centers should be incorporated into the allocation structure and the total cost for each service calculated. Second, the "standard rate" for each service unit should be calculated. The standard rate can be calculated in one of three ways: (1) by dividing the total estimated cost of a service for the planning period by the total projected usage, (2) by dividing the total estimated cost of the service for the planning period by total available DP capacity, or (3) by using a technique that considers the total projected usage and cost over the projected (multiyear) life of the service. Third, coefficients for each charging algorithm variable, if charging algorithms are being used, should be calculated. Fourth, any factors for normalizing charges between competing or similar services, such as multiple machines or DP facilities, should be determined. Fifth, any surcharges, priority charges, and discounts that are to be used should be determined. The end result of these tasks is the calculation of a billing rate (price) for each service offered by the DP facility.

"Step 8" Conduct DP Management Accounting

Step 8 consists of three main tasks. First, procedures should be developed for the establishment and maintenance of user and DP facility accounts. Second, procedures should be developed and implemented for assisting users and the DP facility in establishing and justifying their budgets. Third, procedures should be developed for providing coordination with the organization's accounting and budgeting divisions, the DP facility, and the users concerning areas connected to the charging system, such as the recovery of charges and user accounts. These procedures are needed to keep track of actual expenditures of both the DP facility and the users. FGAP 4 provides the guidance needed for management accounting for the DP facility.

4.4 Billing Phase

Billing includes functional procedures for accounting for usage, producing and distributing usage reports and bills, and recovering charges from users. During the billing phase, these procedures are defined in detail, developed, implemented, and used in an operational environment. The procedures directly affect the users, the DP

facility, and the organization's accounting and budgeting practices. Representatives from these groups should be involved in the detailed design, development, implementation, and operation. Cost accounting for the DP facility is not addressed in the draft Guidelines document; however, cost accounting does occur in parallel with the billing phase.

"Step 9" Develop and Implement Usage Accounting Procedures

Step 9 consists of two major tasks. First, the usage accounting procedures that are needed to collect the necessary data for the charging system should be designed in detail. These procedures typically include both automated and manual methods. When designing the procedures, the designers should be sure that all of the data have been identified from the planning, design, and rate-setting phases. The second task is to develop, implement, and operate the usage accounting procedures. Development of these procedures will typically include purchasing off-the-shelf software.

"Step 10" Develop and Implement Reporting Procedures

Step 10 consists of two major tasks. First, the reporting procedures should be designed in detail. This consists of determining (1) how to reduce the usage accounting data, (2) how to apply the billing rates, and (3) how to report the usage levels and charges. Second, the reporting procedures should be developed, implemented, and operated. Development of the reporting procedures will typically include purchasing off-the-shelf software and making modifications to it as necessary.

"Step 11" Develop and Implement Recovery Procedures

Step 11 consists of four major tasks. First, the policies of the organization for transferring funds should be researched. Second, techniques for adjusting charges in error should be developed. Third, the plan for recovering funds from the users and accounting for those funds should be designed in detail, developed, implemented, and operated in accordance with the agency's policies and practices. Finally, feedback from the users should be obtained and procedures should be modified as needed.

5. Conclusion

The purpose of this paper is to identify the critical phases, steps, and issues in charging system development and implementation. The approach and issues presented in this paper are not intended to provide a comprehensive, detailed plan for developing and implementing a DP charging system. The topic is too complex and far reaching to be standardized; however, a general approach and best practices can be established.

In setting policies and developing guidelines for charging users for the DP services received, the Federal Government is attempting to shift the responsibility for DP costs from the DP facility to the users. Such a shift raises questions that are addressed in OMB Circular A-121, FGAP 4, and the draft Guidelines discussed in this paper. The underlying theme of these new policies and guidelines is improving the accountability of both Federal DP facilities and their users.

In addition to improving accountability, a charging system can also improve both the DP planning process and the communications between the DP facility and its users. The approach presented in this paper stresses the importance of planning and communications as the fundamental elements for the successful design, development, implementation, and operation of a charging system.

The authors thank Dr. Dennis Conti of NBS and the many other professionals in both Government and commercial organizations who participated in the Guidelines development project and the preparation of this paper.

Bibliography

[1] Bernard, Dan, et al, Charging for Computer Services: Principles and Guidelines, Petrocelli, 1977.

[2] Cushing, Barry E., "Pricing Internal Computer Services - The Basic Issues," Management Accounting, April 1976, 57(10), pp. 47-50.

[3] Dearden, John and Nolen, Richard L., "How to Control the Computer Resource," Harvard Business Review, November-December 1973, 51, pp. 68-78.

[4] "Charging for Computer Services," EDP Analyzer, July 1974, 12(7), pp. 1-13.

[5] "Job Accounting and Chargeback," EDP Performance Management Handbook, June 1980, pp. 2.0.1-2.90.5.

[6] "Standard Costing in Data Processing," EDP Performance Review, June 1981, 9(6), pp. 1-6.

[7] "A User-Oriented Approach to Charge-back," EDP Performance Review, February 1975, 3(2), pp. 1-6.

[8] McKell, Lynn, Hansen, James V., Heitger, Lester E., "Charging for Computer Resources," Computing Surveys, June 1979, 11(2), pp. 105-120.

[9] Nolan, Richard L., "Controlling the Costs of Data Services," Harvard Business Review, July-August 1977, pp. 114-124.

[10] United States Office of Management and Budget, "Cost Accounting, Cost Recovery and Inter-agency Sharing of Data Processing Facilities," Circular No. A-121, September 1980.

[11] Schaller, Carol, "Survey of Computer Cost Allocation Techniques," Journal of Accountancy, June 1974, 137(6), pp. 41-49.

[12] Statland, Norman, et al, "Guidelines for Cost Accounting Practices for Data Processing," Data Base, Winter 1977, 8, pp. 2-20.

[13] United States Government Accounting Office, "Illustrative Accounting Procedures for Federal Agencies," Federal Government Accounting Pamphlet Number 4, 1978.

[14] Zmud, Robert W., "Selecting Computer Resources for Inclusion within a Pricing System," Journal of the American Society for Information Science, November-December 1975, pp. 346-348.

CPEUG81

Control of Information Systems Resource Through Performance Management

SESSION OVERVIEW

CONTROL OF THE INFORMATION SYSTEM RESOURCE THROUGH PERFORMANCE MANAGEMENT

John C. Kelly

Datametrics Systems Corporation
Burke, VA 22015

Performance evaluation has expanded during the past few years to take on a more global view of the data processing function. Performance analysts can no longer limit their activities to system tuning. The issues have become broader than just the central hardware and software. Thus the shift in terminology from performance evaluation to performance management. Two trends have forced this change in emphasis. First, with the thrust toward on-line systems, the performance of systems has become visible to a wider and wider user community. Performance evaluation can no longer be confined to the back room. It is out in the open whether we like it or not. Second, the productivity of the entire organization depends more and more on the computer system. In many cases, corporate profits are directly related to computer performance.

The papers presented in this session address some of the issues associated with the trend away from evaluation toward management. David Vincent discusses some of the basic issues associated with performance management and describes how the pieces fit together. Dennis Norman discusses the issues as they relate to an on-line network. The session will finish with a panel discussion of the issues presented in the papers as well as answers to the following questions:

- What is the proper role of computer performance management (CPM)? Does it properly include such activities as programmer productivity?
- Where should CPM be located within the organization?
- How many people are required to support CPM?
- What experience and education are needed for CPM analysts?
- Is it possible to cost justify CPM?
- Is there really a difference between performance evaluation, performance management, and capacity planning?
- How does one take into account user perceptions versus reality?
- What are the most useful measures of performance?

The panel will consist of myself, the two speakers, and the following individuals: Larry Greenhaw of the Texas Department of Human Resources, and Ken Moore of the National Bureau of Standards.

INCREASING SYSTEM PRODUCTIVITY WITH OPERATIONAL STANDARDS

David R. Vincent

Boole & Babbage, Inc.
Educational Services Division
Sunnyvale, CA 94086

ABSTRACT

The user service levels achieved by a data center are dependent on three variables: volume, mix, and efficiency.

By isolating and tracking these elements, the data center can assign accountability to the user for their demands on the system as well as isolate system inefficiencies to be corrected by the data center. These methods also help to create the environment where data center and user can agree on the essentials of the delivery of computer services.

One of the most outstanding individuals to come out of the American Industrial Revolution was a young man by the name of Frederick W. Taylor. Taylor's "scientific management" had a large effect on the tremendous surge of affluence we have experienced in the last seventy years which has lifted the working masses in the developed countries well above any level experienced before, even that of kings and queens of old. Taylor's analysis of work and resultant method improvements resulted in productivity gains unequaled in history[1].

His analysis of work in the 1880's was exemplified by his study of shoveling iron ore in a steel mill. In this study, he found gross inefficiencies in the actual process of shoveling. By breaking the motions and actions of the workers down into measurable segments, he was able to develop better work methods (something like tuning a computer) and standards of performance so that output could be judged on a day-to-day basis.

This work was finally completed, as we now know it, shortly after he passed on during World War I, at which time American industry in general began to adopt his principles. Some of Taylor's disciples carried on his work thereafter; notably, Frank and Lillian Gilbreath (who were the subject of a movie "Cheaper by the Dozen") and Henry Gantt, who initiated project scheduling in a Gantt Chart. The common interest uniting those people was the analysis of work and translating that analysis into more productive and efficient procedures and flows of work.

The analysis of work involves the following:

1. The identification of all processes necessary to produce an end product or result
2. The rational (and manageable) organization of the sequence of operations so as to make possible the optimal flow of work
3. The analysis of each individual operation or process including measurement and historical trending
4. The integration of the above into an overall process of producing a product or result

This established process has been used successfully for decades in American industry (and Japanese, and German, and . . .). It is my contention that the size and complexity of the data center has now evolved to a point where this methodology may be applied in greater depth to realize significant economic benefits.

---

[1]Actually, Taylor was preceded in philosphy, by another "irascible genius", Charles Babbage, who documented some of the earliest forms of scientific management in his most successful book, On the Economy of Machinery and Manufactures (1832). However, it took Taylor to "rediscover" and implement practical scientific management in America.

The role of the data center in the organization to which it belongs has increased from simple payroll and accounting applications in the 60's (remember "tab runs"); through inventory and distribution systems in the 70's; to online, realtime management and operational applications in the 80's. A good example of the online operational application of the 80's is the automated teller systems in banks. In this case, the data center has gone to a point where it actually interacts with bank customers.

As the data center evolves towards a utility-like process, the end product of the data center is service. With the proliferation of online systems, service has become highly visible to the user and a much more tangible element in top management consideration. In fact, service levels have become so visible to the world outside the data center that, to a large extent, they are considered the measure of the data center's performance. The perception of service falls into three basic categories[2]:

1. Online response time
2. Batch turnaround
3. Availability

When one of these is outside a user's expectation, the DP manager's phone begins to ring with complaints. Moreover, the business environment in which the DP manager now lives is one that expects him to manage his operation the same as any other functional unit of the enterprise. At this point, many DP managers are beginning to feel the strain of trying to negotiate and maintain service levels without the benefit of having fully implemented traditional scientific management principles in their data centers.

One major element of scientific management is the need to understand the elements of the service to be performed and the variables that can affect them. These variables have one of two sources: data center operation or user behavior. There are many data centers that have not yet quantified the difference in, say, response time caused by data center inefficiency as opposed to that caused by user behavior. This is because the analysis of work has not been related to these two factors.

These data centers are usually perceived by its users as being poorly run because all variances from negotiated response times are attributed to the inefficiency of the data center. When analyzing work, there are in fact only three universal causes of deviation from a standard which will cause response time to be better or worse than plan:

1. Volume
2. Mix
3. Efficiency

The financial term for these deviations is variance and simply means the difference between planned versus actual results. The first two variances are attributable to user behavior. The third variance is the only one that can be attributed to the operating of the data center. In essence, the planning and anticipation of the volume and mix considerations (user behavior) are much of what capacity management is all about.

In this paper, I will explore methods of determining the relationship of volume, mix, and efficiency on the current system and how we might predict performance at various levels. The goal will be to define performance curves giving standard levels of service at varying levels of volume and mix.

Volume is a measure of activity in terms of jobs or transactions by job or transaction type. The various jobs or transactions in types result from the differences in computer resources consumed to process that particular job or transaction. Differences in volume can be measured by comparing forecast or plan to actual. For example, the running of Job A123 resulted in the following volumes:

JOB	PLAN TIMES RUN	ACTUAL TIMES RUN	VARIANCE (PLAN-ACTUAL)
A123	4	6	<2>

The volume variance in this case is expressed as 2 jobs over plan. In terms of resources expressed in standard work units[3], it is 2 times planned resource per run of Job A123. Let's say Job A123 should use:

[2]These were the first three items listed in a poll done in Arizona with 150 DP managers. Appendix A contains a full list prepared during the 1981 EDP Performance Management Conference.

[3]For this example, a standard work unit will be considered for the case of CPU only. The standard work unit is basically CPU time factored for the relative power of the CPU.

	Units (000)
CPU standard work units	= 170
I/O	*
Memory	*
Total standard work units	= 170

Therefore, the volume variance is 2 times 170,000 SWU's units or 340,000. This may also be expressed in terms of dollars if a standard cost per SWU's can be calculated and applied.

Now let's take an example where there is a volume and mix variance. Let's suppose that we have 22 jobs that require a total of 4,320,000 SWU's (an average of 196,364 per job).

The actual activity turns out to be 24 jobs requiring a total of 5,760,000 SWU's for a variance of 1,440,000. The matrix of the above with additional information is then constructed as follows:

	WORKLOAD			(000) STANDARD	TOTAL SWU's (000)		
JOB	P	A	V	WORK UNITS	P	A	V
A123	4	6	<2>	170	680	1020	<340>
B357	8	3	5	80	640	240	400
C896	10	15	<5>	300	3000	4500	<1500>
Totals	22	24	<2>		4320	5760	<1440>

The volume variance is calculated:

Variance of job runs X $\frac{\text{Planned SWU's}}{\text{Planned Workload}}$ = Volume Variance

The volume variance is calculated:

Variance of Job Runs

X

$\frac{\text{Planned SWU's}}{\text{Planned Workload}}$

= Volume Variance

1. The variance of job runs is calculated by subtracting the actual total jobs run (24) from the plan (24) with a resultant 2 jobs run over plan. The brackets indicate that this variance will have an unfavorable impact on data center performance.

2. Planned SWU's divided by planned workload is in fact the planned rate of SWU consumption for a job from this workload.

3. The volume variance is then a function of the jobs run over or under plan times the planned resource (SWU) consumption rate. This isolates those resources consumed over or under plan as a result of users running a different number of jobs than planned.

4. The numeric calculation of volume variance is then:

$$\langle 2 \rangle \times \frac{4320}{22} = \langle 392.8 \rangle$$

The mix variance is calculated[4]:

1. The planned SWU rate ($\frac{\text{Planned SWU's}}{\text{Planned Workload}}$) was just calculated for use in the volume variance.

2. The actual SWU rate is the key element in the calculation of the mix variance because the difference between the planned and actual SWU rates is a result of a workload with different elements.

3. The difference in the SWU rates is multiplied by the actual workload.

4. The numeric calculation of this example is then:

---

[4] In this case, the SWU's per job are not varied. As can be seen in the next example, they are varied and this will result in an efficiency variance. The formula for calculating mix variance would be modified to reflect the fact that both mix and efficiency variance contribute to the difference between the planned and actual SWU rate. The revised formula would then be:

$$\frac{\text{Planned SWU's}}{\text{Planned Workload}} - \frac{\text{Actual SWU's}}{\text{Actual Workload}}$$

X

Actual Units

Efficiency Variance = Mix Variance

= MIX VARIANCE

$$\left(\frac{4320}{22} - \frac{5760}{24}\right) \times 24 = \langle 1047.2 \rangle$$

The mix variance is the amount of standard work units consumed over or under plan as a result of having different jobs run than were planned which, in turn, consumed different amounts of resources. The summary of the variance is then:

Volume Variance	<392.8>
Mix Variance	<1047.2>
Total Variance	<1440.0>

Now we can expand the same example to demonstrate an efficiency variance. Let's say that we have the following data:

	Workload			Standard Work Units (000)		
Job	P	A	V	P	A	V
A123	4	6	<2>	170	165	5
B357	8	3	5	80	100	<20>
C896	10	15	<5>	300	500	<200>
Total	122	24	<2>			

Total SWU's (000)		
P	A	V
680	990	<310>
640	300	340
3000	7500	<4500>
Total 4320	8790	<4470>

In this case, the SWU''s consumed per job were different than planned. Since we have kept everything the same except the actual SWU's per job, the volume and mix variances are exactly as previously calculated.

The efficiency variance is calculated by:

(Planned Job SWU_1 - Actual Job SWU_1)
X
Actual $Jobs_1$ = Efficiency $Variance_1$
+
(Planned Job SWU_n - Actual Job SWU_n)
X
Actual $Jobs_n$ = Efficiency $Variance_n$

1. The total efficiency variance is a sum of the individual efficiency variances calculated for each job run.
2. Each job variance is calculated by multiplying the actual runs of the job times the job's SWU variance (planned minus actual SWU's to process each run of the job).
3. The calculation would then be:

Job	Actual Times Job Ran	SWU Variance	Efficiency Variance
A123	6	5	30
B357	3	<20>	<60>
C896	15	<200>	<3000>
Total Efficiency Variance			<3030>

The new total variance summary is then:

Volume Variance	<392.8>
Mix Variance	<1047.2>
Efficiency Variance	<3030.0>
	<4470.0>

This says that the data center performance of running these jobs, especially Job C896, suffered either because of:

1. A bad application program,
2. Some system deficiency,
3. A bad estimate of what it would take to run the job, or
4. A bit of all three.

It should be noted that user behavior caused 32% of the total variance. Often, the user behavior element contributes even more to the total variance, especially at peak periods.

Where can SWU's be obtained? Most computer systems have some kind of log that accumulates resource usage. CPU time is the easiest measure, but relative CPU power differences dictate that adding pure CPU time from different CPU's might be erroneous over time. IBM has offered a solution with MVS by providing Service Units which are internally calculated. These Service Units are indeed CPU time times an internal power factor which result in theoretically compatible service units over a variety of IBM CPU's.

The Institute for Software Engineering has also provided a great deal of literature in this area which deals with Software Physics. The direction of this work is somewhat similar to the service unit methods built into IBM and PCM systems, but is much more complex.

So far, the analysis of variance has focused on service units or the amount of work going through a data center. How do these translate to levels of service?

The data processing work plan basically consists of putting out the required volume and mix of work as a basic requirement, and on a timely basis as a second but equally important requirement in most shops. We can be sure that capacity has been exceeded when the data center physically cannot process the required workload even if it were to run a full three shifts per day, seven days a week. But below this level, there are other considerations that become practical limitations of available hours to do data processing work. The variances analyzed earlier in this paper will affect the timeliness of the data turnaround, especially in on-line systems. Plans of user workload are especially important during the peak online requirement that occurs during the normal five-day work week. This is especially important if the data center is very involved in the basic business of the company with which it works, such as a bank or department store.

What we're really talking about here are the end user's work schedules and how that impacts the ability of the data center to plan and deliver services matching their schedules as postulated in the User Behavior Elasticity Theorem[5]; which states that the degree to which the data center can influence end-user behavior is inversely proportional to the degree that data processing is involved in the basic business of the organization. This can be illustrated by two examples.

The first is in the banking industry where automated tellers are being implemented. The data center cannot influence the end-users of this application to any noticeable degree because the bank's basic business depends upon having the automated tellers operating when its clients (i.e., depositors, withdrawers, etc.) want to make a transaction. Chargeback schemes, management pressure, and the like will have little or no effect. On the other hand, a company producing buggy whips that has not integrated data processing into its basic business will tend to be much more flexible in terms of end-user behavior because production and distribution will continue whether the data center runs or not. Hence, it is more elastic as charging schemes and management pressure are applied.

We are also talking about the necessity for conscious management decisions regarding the economic benefits of achieving a given online response time versus the system costs that will be required to provide that level of response. Or, as another case, the adding of another application to the online system in light of its potential impact on the response time of current users and applications may or may not be possible under the current configuration because of the impact it will have on the service required for other applications. This is where performance management comes in, specifically, the analysis of performance data. What we want to know is which user-controlled variables, i.e., volume and mix, will impact service level performance. Those of us who have been tracking this type of data over time know what happens when a CPU gets over 90% busy and are well aware of exponential degradation of service levels that occur beyond this level. This is also true for many other areas within the system depending on where the system bottleneck is. Are there tools that will identify such sensitive areas in the system? If so, how might they be applied?

The software tools and sources of data needed to perform this kind of analysis are readily available. You probably have some of them installed on your system already.

When the data center has analyzed the effects of user behavior and workload, the next logical step is to relate them to the capability of the system. Peak period performance will probably be the main ingredient in any service agreement. The first step in predicting system capability is to know what the system can do now. By implementing performance reporting, there will be data relating to user workload characteristics. These can then be related to service level achievement.

First, we must assume that the system is properly tuned. A second assumption is that the workload has been shifted to the extent possible (i.e., batch work at night so as not to interfere with online work). At this point, the theorem of User Behavior Elasticity applies. The next logical step is to develop the operational standards of performance in terms of service levels based on the current configuration and end-user service level objectives.

---

[5]This is my theorem developed from personal observation and many discussions about chargeback systems.

In the case of predictive models, such questions as "What effect on service levels will be experienced by adding another channel or DASD device?" can be modeled and the results calculated against the current workload. Future anticipated workload growth can also be modeled to see future service level achievement with the current system. Shifts in volume and mix as compared to plan, when modeled, will illustrate service degradation caused by user behavior.

Alternative hardware and software options may be considered to find what is needed to maintain negotiated end-user service level requirements. Additions to the current system or other alternative systems may also be modeled to measure the impact.

The major characteristics of a good predictive model are the following:

1. Results can be easily validated
2. Easy to use
3. Will distinguish the various hardware and software characteristics of available products
4. Can easily integrate user historical data to define workload characteristics
5. Economical to run
6. Easy to interpret reports

The first and foremost requirement is that the model be easy to validate. The value of a predictive model is in its prediction!

The second and very important requirement is that the model be easy to use. This will reduce the need for highly technical systems people or, at the very minimum, a mathematical theoretician to use the product. Rather, the model should be usable by a trained business analyst. Most errors made by models are created from erroneous input. Complicated models tend to be resplendent with such opportunities for error.

The predictive model ideally will easily incorporate hardware and software alternatives available to the data center, so that the analyst can play "what if?" games. That is, various configurations can be matched against various workload and service-level requirements to determine which configurations would be optimal. This type of analysis lends itself extremely well to the decision-making needed in the process of appropriating capital goods (i.e., data center hardware or software). From this data, various suitable configurations can be selected and the cost effectiveness of each evaluated. The cost of various user service levels may also be calculated. Both the capital and service analysis will assist in establishing the financial requirements for the data center.

It is important that the model be fed from an historical data base fed by the various system monitors so that workload data can be easily integrated. This may be used to define current standards of performance as well as defining trends for predictive modeling.

The model should be economical to run, that is, it should not consume much computer time. Since this process will be interactive and many passes of the model will be required to determine the new performance curve, each pass should run in a couple of minutes or less.

And finally, the output must be easy to interpret and readily presentable for management reporting. Again, a business analyst should be able to interpret the output and be able to input various alternatives as a result of the output from any given pass of the model.

The objective of this measuring, analyzing, and modeling will be to derive performance curves that can become operational standards of performance that show the relationship of service level achievement versus user behavior. Exhibit 1 shows the process involved in establishing such standards. It is an interactive process that involves tuning and end-user negotiations until finally an agreed standard of performance has been set. However, the standard is dynamic. Exhibit 2 is a generalized performance curve with user-controlled variables along one axis and service level performance along the other. User-controlled variables are, in fact, the various levels of volume and mix for each of the various categories of work.

In this case we will discuss TSO transaction response as a function of the volume of user activity. The data for this graph may be obtained from CMF, RMF, or SMF. In the case of a DOS or in non-IBM environments, the system log that records the volume of activity and system resources consumed should provide the necessary data. Exhibit 3 is an example of a System Workload Summary[6] from which the following data can be obtained:

---

[6]Exhibits 3, 4 and 5 were produced by CMF for an IBM MVS system.

1. Volume of transactions by performance groups
2. Service units used in each system area

Exhibit 4 is the CPU Utilization Report which shows:

1. CPU busy data
2. CPU queue data

Exhibit 5 is a TSO Subsystem Performance Report which shows:

1. TSO response by time period
2. TSO response by command
3. Concurrent TSO users

The above data was fed into the SAS statistical program to determine which data showed a relationship between TSO response and another variable. The variable with the closest relationship turned out to be the CPU queue time[7] which is directly affected by user volume and mix. In the case of our system, we are currently CPU-bound, so this is not a surprising bottleneck. As you can see in Exhibit 6, a linear regression line has been drawn with the standard error shown as a dotted line on either side. In this case the standard error is 1.1 seconds on either side of the regression line. Basically this says that two-thirds of the time, observances of TSO response versus CPU queue time will fall within the area bounded by the dashed lines.

The same analysis was done for TSO mix in Exhibit 7. This line turned out to be flat and was essentially meaningless because of the variation in data. Again, this corresponded to what we wish the system to do. TSO has a high priority, so even at high CPU busy levels, the TSO service should not suffer.

Exhibits 8 and 9 show the effect on TSO response caused by total service units consumed and concurrent users respectively. In both cases there is a direct relationship, even though the standard error is larger. As you can see, using historical data, system interrelationships with user behavior can be derived. However, this kind of data tends to be linear, and does not answer the question of how the system will react to user behavior at levels not yet experienced.

A third problem is that the system is never exactly the same from one period to the next. For this reason, it is extremely important to correlate performance data from the system to data logged from a change management tracking system. A change management tracking system is basically a problem reporting system for all hardware, software, and applications problems and changes made to the system. There have been many times that a one-byte code change on Sunday night caused a system to come to its knees on Monday morning with resultant days of analysis before the change was found. Each system change needs to be documented and available for analysis when performance data shows a major deviation. This analysis goes a long way towards explaining the efficiency variance we discussed earlier and should be mapped to each change in the performance of the system.

Using a Model to Plot a Curve

Because graphing historical data will not always answer the questions of future system behavior for future user workloads, a model is often used to simplify the process. By using a model, curves may be defined for each system limitation as well as an overall curve for the present system capability. This requires many iterations of a model to define the points on the curve and even more iterations to define other possible curves. It is also important to validate the model to actual system results. That means, if we pick a point on the performance curve and take the corresponding volume and response, how close does this match reality? In other words, will the point fall within the bounds defined in the linear projection and standard errors graphed in the analysis of historical data as was shown in Exhibit 8?

If indeed the model can be validated and the results are consistent, we have in fact defined an "operational standard." "Operational standard" as used here means that there is a standard performance characteristic for a given level of user activity on the current system. This becomes an extremely powerful tool for negotiating service levels with users. A major misunderstanding with users can be avoided if they realize that for some levels of user activity, response will be lower. This is especially true if there are peak periods with extreme activity for short periods during the day and that activity causes the "knee" of the curve to be reached.

Exhibit 10 comes from an actual case where a factory made a union agreement to clock out all employees in ten minutes.

---

[7] As calculated by Little's Rule, $L=\lambda Q$, or the mean length of a queue is equal to throughput times queueing time.

This agreement brought the system to its knees. The exhibit shows a performance curve for a 370-148 which averages 2.0-second response time for about 10,000 transactions during an 8-hour period. If, however, 625 transactions come in between 4:00 and 4:10 and must be processed in the same two-second response time, a different system will be required. This is due to the fact that 625 transactions in a ten-minute period are equal to 30,000 transactions in an 8-hour period. A curve is drawn for a 3033 to illustrate the system expansion needed to accommodate the 10-minute traffic at 2-second response. In this case, the company management will have to weigh the service requirement against the additional investment. If user behavior is relatively inelastic, there will soon be a 3033 installed.

Exhibit 11 is a performance curve of the Boole & Babbage data center that was made as an example for a case study. This system is comprised of:

CPU	M80 (370-148) plus 6MB memory
Disc	6x 3330
	8x STC 3630 (3350)
Tape	3x STC 4534
Channels	4 plus byte multiplexor
Operating System	MVS/JES2/TSO/SPF/VAM/CICS

Response time shown by the model degraded significantly after 20 concurrent users and 1.1 transactions per second. In further analysis of this case, we found that we are indeed CPU-bound. This means that as the CPU resource is consumed by a workload or variations in user behavior patterns, degradation of service will be a direct result. A 4341 Model 2 with expanded CPU capacity is on order and we hope to see some relief this fall when it is installed. The next step in this process will be to model the performance curve of the new system. We may then find some other system bottleneck such as I/O or DASD. In the meantime, we now have our operational standard of performance for TSO.

The Future of Performance Curves

The modeling of a workload against a given system is not only necessary, but is feasible with currently available tools and technology. The computation of and comparison to planned performance curves has value in determining:

1. Data center efficiency
2. The effects of user behavior (volume and mix)
3. Benefits of tuning
4. The ability of the data center to meet various levels of activity
5. The benefits of various system alternatives

Because of these benefits and the fact that future models will get even more involved in simulating operating system parameters (i.e., SRM under MVS), performance curves should be available on a real-time basis. This means linking the Change Management Tracking System and realtime system monitor/model so that early warning mechanisms can be implemented. Realtime monitors will in effect simulate system changes and signal when the performance curve has changed from plan. This will provide an effective tuning tool by displaying these changes much like the IPS parameters in which domains and multiprogramming levels are displayed in the IBM Initializaion and Tuning Guide.

By monitoring the effects of user behavior, both on a continuous after-the-fact basis as well as in models, the data processing operation has implemented an important element of scientific management. The work analysis is done by the system itself while the manager deals with the question of user behavior and the integration of DP into the business.

APPENDIX A

MEASURES OF PRODUCTIVITY IN DP OPERATIONS

RESULTS OF VOTING*

MEASUREMENT FACTOR	RANKING	VOTES
ONLINE RESPONSE TIME	1	198
ON-TIME REPORTS	2	160
SYSTEM UPTIME	3	141
USER SATISFACTION	4	137
RERUN PERFORMANCE	5	101
REPORTS DISTRIBUTED W/O MISTAKES	6	56
NUMBER OF INTERRUPTS IN ONLINE SERVICE	7	55
PROBLEM RESOLUTION TIME	8	47
CPU UTILIZATION	9	45
COST OF OPERATION	10	43
ACTUAL VS. SCHEDULED RUN TIME	11	39
DEMAND BATCH TURNAROUND	12	34
RECOVERY TIME FROM FAILURE	--	34
TIME SHARING INTERACTIVE RESPONSE	--	30
LATE JOBS FAULT OF OPERATOR	--	30
OUTAGES BY CATEGORY	--	29
APPLICATION PROGRAM ABENDS	--	27
NUMBER OF PROJECTS WITHIN BUDGET	--	24
HARDWARE/SOFTWARE RELIABILITY, INTFAC	--	23
NUMBER AND TIME OF TAPE MOUNTS	--	20
JOBS COMPLETED PER COMPUTER HOUR	--	19
PEOPLE TURNOVER	--	16
ABSENTEE RATE	--	15

*Based on a survey of a meeting of data center managers at the 1981 EDP Performance Conference in Phoenix, Arizona, February 23-26, 1981.

APPENDIX B

EXHIBITS

EXHIBIT
1. Establishing operational standards
2. Generalized performance curve
3. CMF system workload summary
4. CMF CPU utilization report
5. CMF TSO subsystem performance report
6. Linear regression - TSO response/CPU Q
7. Linear regression - TSO response/TSO mix
8. Linear regression - TSO response/SU's
9. Linear regression - TSO response/concurrent users
10. Performance curve for B/B DC
11. Performance curve for B/B DC

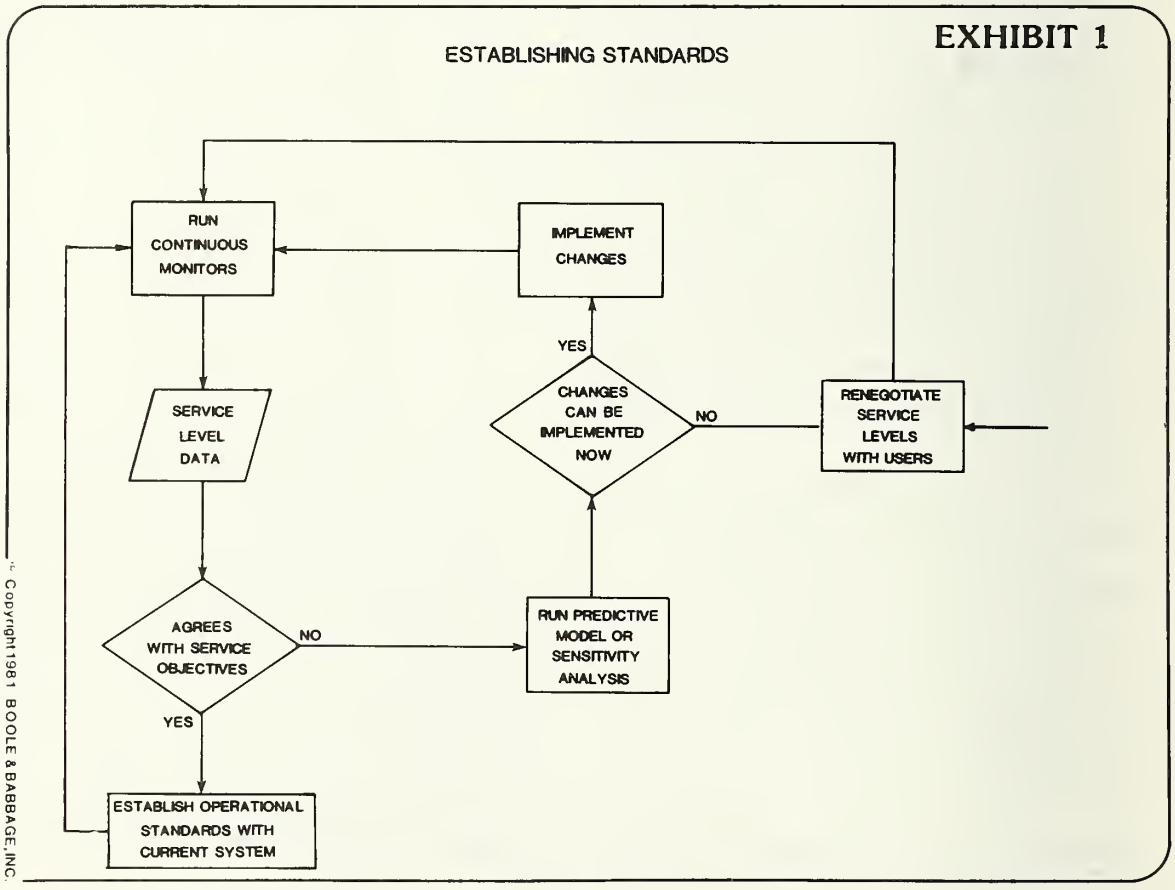


EXHIBIT 2

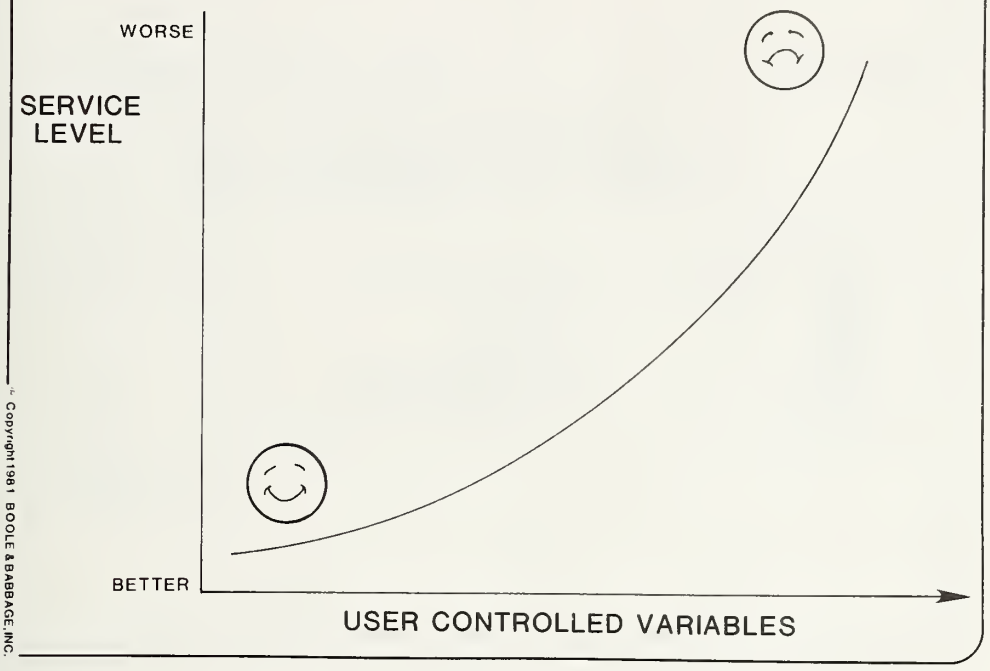


EXHIBIT 3

SYSTEM WORKLOAD ACTIVITY REPORT

SERVICE USAGE BREAKDOWN BY PERFORMANCE GROUP

PERFORMANCE GRP/PER/SYS	TCB SERVICE SU/1000	TCB SERVICE PCT	SRB SERVICE SU/1000	SRB SERVICE PCT	I/O SERVICE SU/1000	I/O SERVICE PCT	MEM SERVICE SU/1000	MEM SERVICE PCT	TOTAL SERVICE SU/1000	TOTAL SERVICE PCT	SERV RATE	ABS RATE	SWAP COUNT	ENDED TRANS	AVG TRX TIME HH.MM.SS.TTT
PERFORMANCE GROUP 2															
2/1 /00	158.0	77.9	N/A	N/A	7.7	19.3	48.0	23.4	214.2	25.4	18.0	14.4	227	972	0.00.04.125
2/2 /00	14.6	3.3	N/A	N/A	.4	1.5	5.1	2.1	24.4	2.9	19.3	15.4	57	203	0.00.07.710
2/3 /00	15.2	2.7	N/A	N/A	.0	2.3	3.4	1.4	13.6	2.3	26.3	21.0	17	49	0.00.21.336
2/4 /00	11.0	2.1	N/A	N/A	1.7	4.2	2.0	.9	15.7	1.9	10.5	8.4	5	11	0.00.85.132
SUMMARY	203.0	36.0	N/A	N/A	10.0	26.8	58.1	24.9	273.9	32.4	17.8	14.2	306	1235	0.00.06.120

EXHIBIT 4

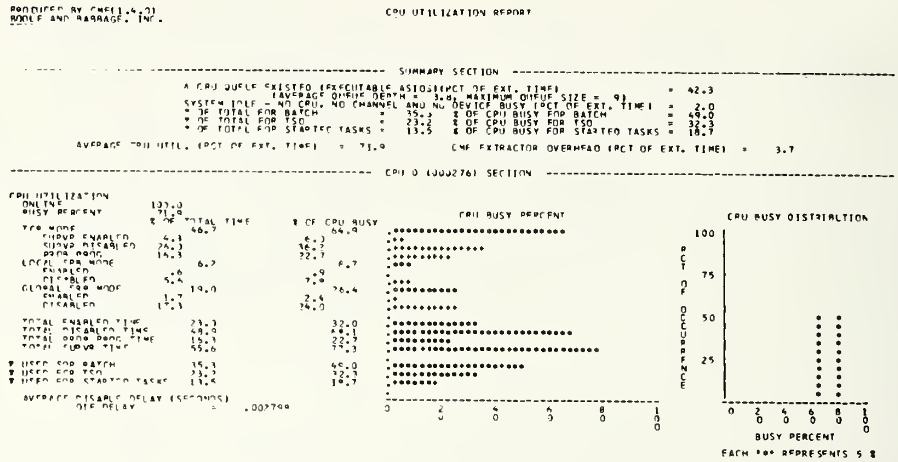


EXHIBIT 5

EXHIBIT 6

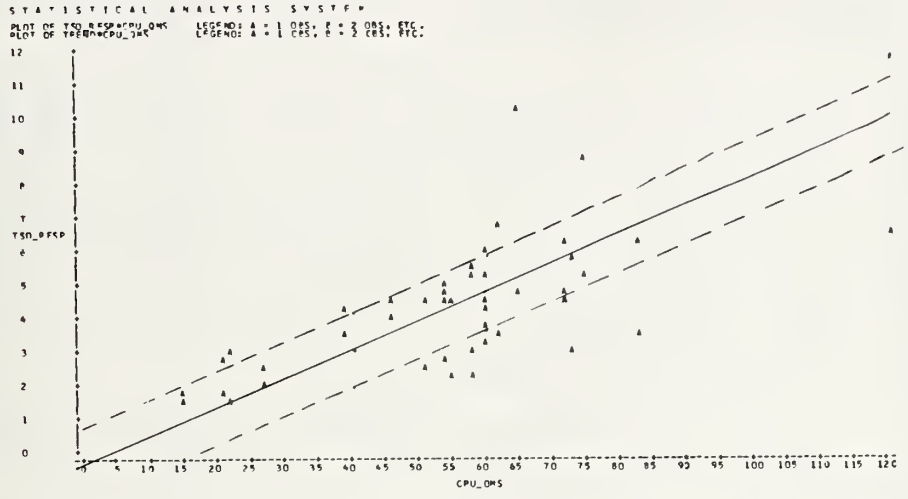


EXHIBIT 7

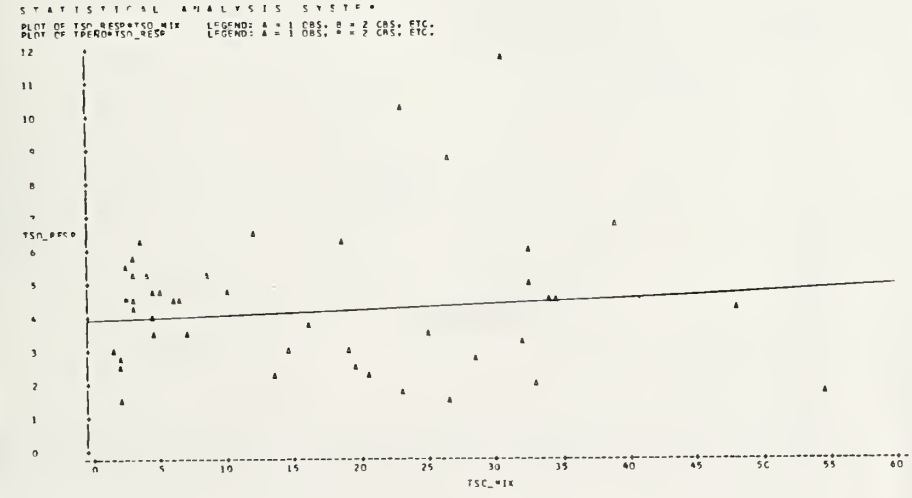


EXHIBIT 8

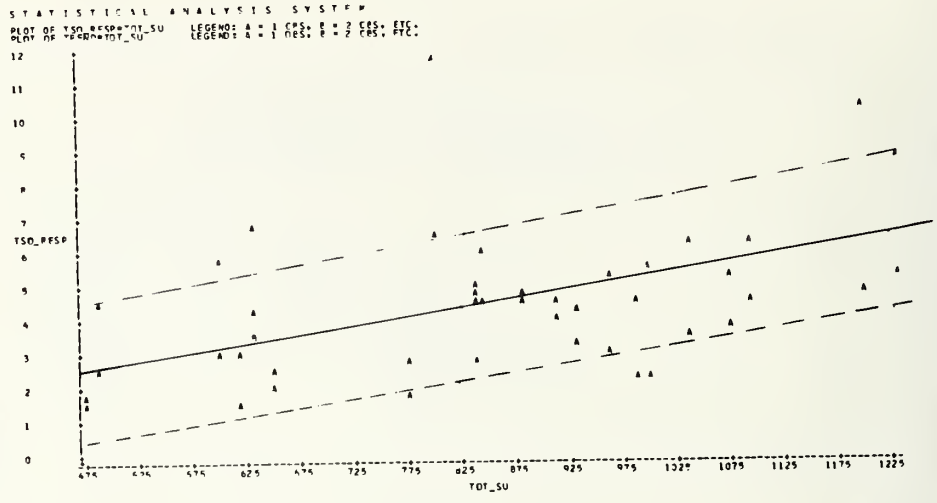


EXHIBIT 9

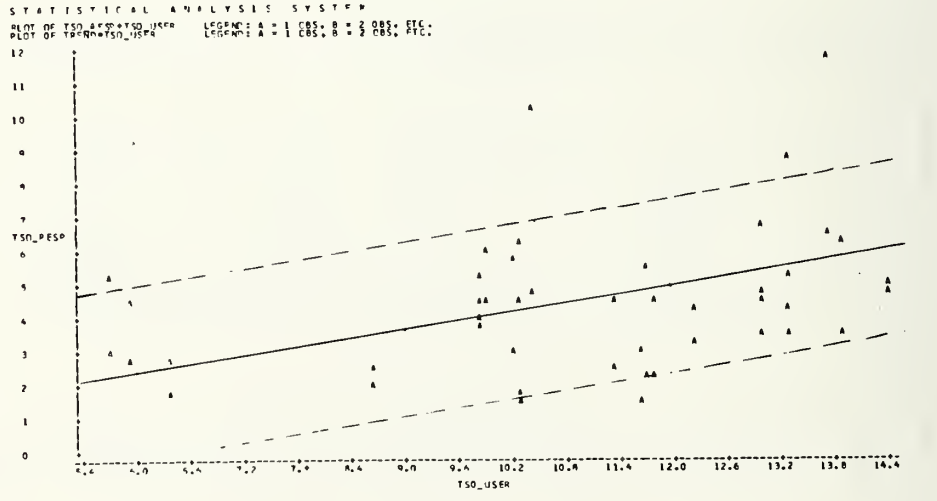


EXHIBIT 10

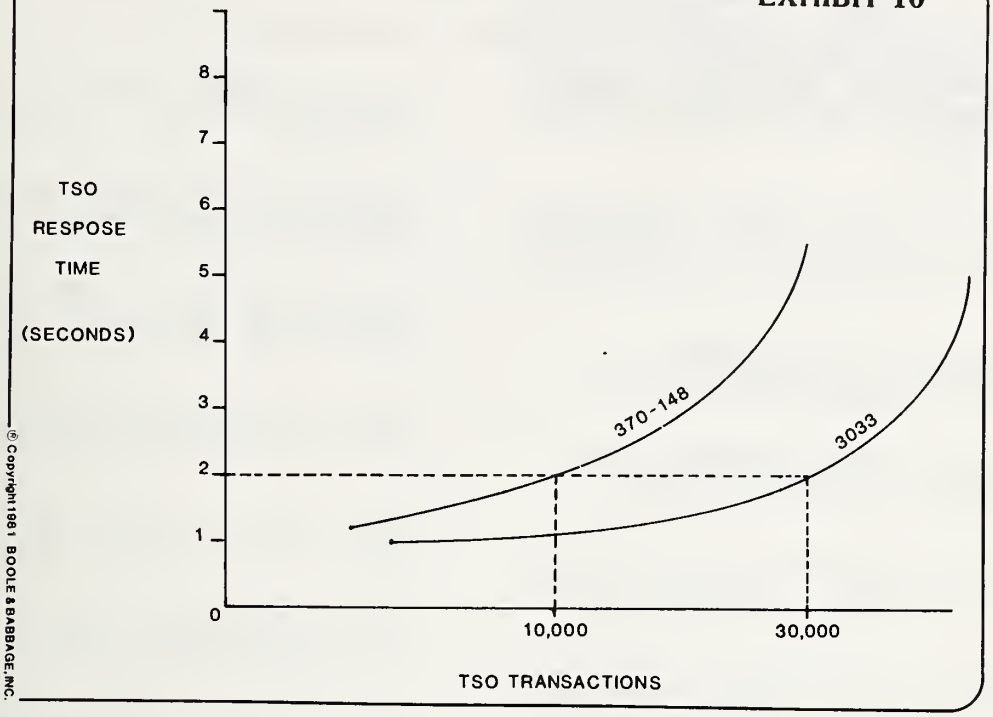
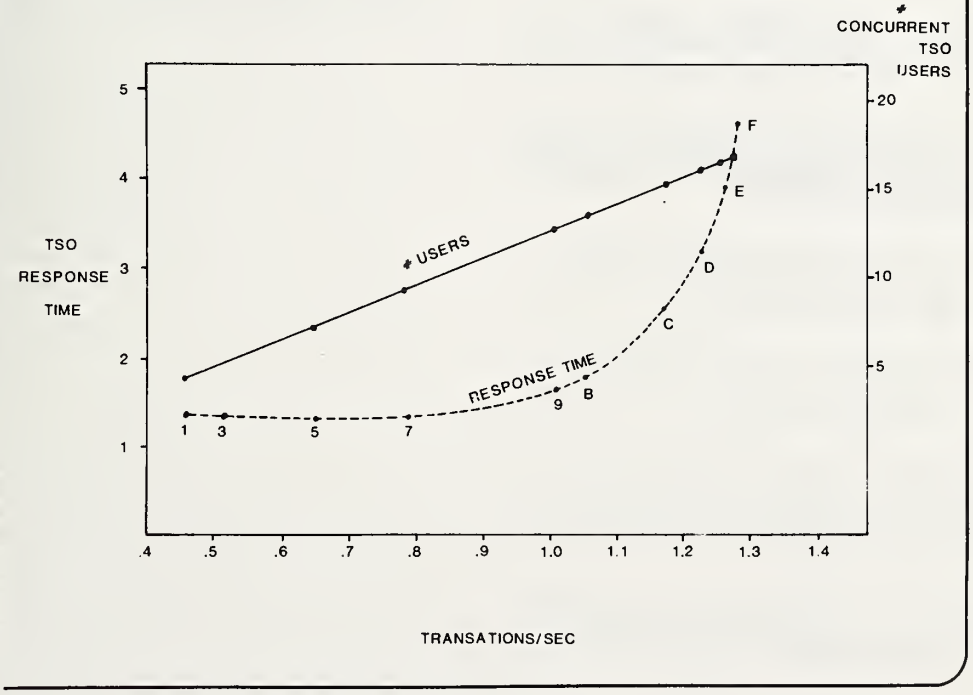


EXHIBIT 11



APPENDIX C

A BIBLIOGRAPHY FOR FURTHER STUDY

1. Arnold O. Allen, Probability, Statistics and Queueing Theory with Computer Science Applications, Academic Press, New York, N.Y. (1978).

2. C. Warren Axelrod, Computer Effectiveness: Bridging the Management/Technology Gap, Information Resources Press, Washington, D.C. (1979).

3. L. Bronner, Capacity Planning: An Introduction, IBM Technical Bulletin GG22-9001-00 (January 1977).

4. L. Bronner, Capacity Planning: Implementation, IBM Technical Bulletin GG22-9015-00, (January 1979).

5. J.P. Buzen, "Queueing Network Models of Multiprogramming", Ph.D. Thesis, Harvard University, Cambridge, Mass. (1971).

6. Computer, April 1980, IEEE Computer Society (contains several articles of interest).

7. Peter F. Drucker, Management: Tasks-Responsibilities-Practices, Harper & Row, New York, N.Y. (1974).

8. Jeffrey L. Forman, Change Communication: A Management System, IBM Technical Bulletin GG22-9254-00 (July 1979).

9. An Architecture for Managing the Information Systems Business, Volume I: Management Overview, IBM GE20-662-0 (January 1980).

10. Problem and Change Management in Data Processing - A Survey and Guide, IBM GE19-5201-0 (August 1976).

11. IBM System Journal, Volume Nineteen, November 1, 1980, "Installation Management, Capacity Planning."

12. H. Kobayashi, Modeling and Analysis: An Introduction to System Performance Evaluation Methodology, Addison-Wesley, Reading, Mass. (1978).

13. J.D.C. Little, "A Proof of the Queueing Formula, L= W, "Operations Research 9, pgs. 383-387 (1961).

14. J. Martin, Design of Real-Time Computer Systems, Prentice-Hall, Englewood Cliffs N.J. (1972).

15. J. Martin, Systems Analysis for Data Transmission, Prentice-Hall, Englewood Cliffs, N.J. (1972).

16. Montgomery Phister, Jr., Data Processing Technology and Economics, Santa Monica, Calif. (1977).

17. Charles H. Sauer/K. Mani Chandy, Computer Systems Performance Modeling, Prentice-Hall, Englewood Cliffs, N.J. (1981).

18. David R. Vincent, "Software Tools for Service Level Management", Data Management, pgs. 25-29, (March 1981).

19. David R. Vincent, "Measuring Performance Online", ICP Interface, Data Processing Management, (Summer 1980).

20. David R. Vincent, "Service Level Management", 1980 CMFGXI Proceedings, pgs. 196-207.

21. Daniel A. Wren, The Evolution of Management Thought, John Wiley & Sons (1979).

Benchmarking

SESSION OVERVIEW

BENCHMARKING

Barbara Anderson

Federal Computer Performance Evaluation
and Simulation Center
Washington, DC

Benchmarking is a method of measuring system performance against a predetermined workload. The approaches and tools used to represent this workload in benchmark tests have been evolving and improving over the last several years. The first two papers in this session; 1) "Universal Skeleton for Benchmarking Batch and Interactive Workloads" by William P. Kincy, Jr., and Walter N. Bays of the Mitre Corporation, and 2) "Comparing User Response Times on Paged and Swapped Unix by the Terminal Probe Method" by Dr. Luis Felipe Cabrera of the University of California, Berkeley and Dr. Jehan-Francois Paris of Purdue University present new approaches to contructing benchmarking tests. The final paper, "Practical Application of Remote Terminal Emulation in the Phase IV Competitive System Acquisition," by Deanna J. Bennett of the Phase IV Program Management Office of the Air Force and the mini-panel with Randy Woolley of the Tennessee Valley Authority and Robert Waters of the Air Force Computer Acquisition Center detail recent experience in various aspects of benchmarking for system acquisition.

UNIVERSAL SKELETON FOR BENCHMARKING BATCH AND INTERACTIVE WORKLOADS: UNISKEL BM

Walter N. Bays
William P. Kincy, Jr.

MITRE
Houston, TX
77058

The most important objective of the acquisition of new computer systems is to acquire the least expensive system or systems which have the necessary capability to meet workload and quality of service requirements.

The determination of "the necessary capability" may be accomplished by several methods; the most important and accurate method being the execution of performance benchmarks in a "Live Test Demonstration (LTD)".

Building benchmarks is time-consuming and costly. This is particularly true if the benchmarks are designed to represent accurately the batch and interactive workloads to be executed.

One solution would be a benchmark which could be adapted to represent any workload requirements relatively inexpensively. Such a benchmark (UNISKEL BM) is the subject of this paper.

1. Introduction

1.1 Background

UNISKEL BM was developed for NASA's Johnson Space Center, Houston, Texas by MITRE. The primary objective of the benchmark effort was to design representative benchmarks which could be used to select new computer systems to replace obsolete systems in the JSC Central Computing Facility (CCF). The CCF is a general purpose, open shop-type computing facility supporting the administrative, shuttle mission planning and shuttle engineering and development functions at JSC.

An additional objective of the benchmark effort was to emphasize, where possible without adding to the cost of the development effort, the universal applicability of the benchmark to any agency having similiar benchmarking requirements. These goals led to the design of a benchmark skeleton which could be used by any agency in custom designing a benchmark. In fact, this benchmark skeleton was used by MITRE to build benchmarks for the the Bureau of Census to be used in their acquistion of the Bridge System.

1.2 Use of UNISKEL BM

The benchmark skeleton described in this paper provides a procedure for integrating programs (SYNTHETIC TASKS) and descriptions of functions to be accomplished by system software (editor, utilities, etc.) (SYSTEM PROCESSOR TASKS) into job streams.

These job streams are further organized into benchmarks representing specific workloads to be executed by the System Under Test (SUT). These synthetic and system processor tasks may be interactive or batch and may be designed to represent a real workload (one being executed on an agency's system today) or a perceived workload (one expected to be executed on new systems in the future).

A model of any real workload may be derived from accounting data. Although this workload inevitably has the characteristics of the system it was excuted on, UNISKEL BM represents it as a mix of vendor-independent FORTRAN or COBOL tasks, and functionally specified system processor tasks. The resulting benchmarks are independent of a vendor's accounting convention and include a mix of tasks representing an explicit amount of work to be accomplished. The work to be accomplished is described by the following:

SYNTHETIC TASKS:

Virtual Storage Size
Number of kernels to be executed
Number of accesses to tape/disks
Record Size
Number of lines of print
Think time (Interactive Tasks)

SYSTEMS PROCESSOR TASKS:

Functions to be performed

The major advantage of UNISKEL BM is that an agency procuring new systems can save the time required to design synthetic programs and a benchmark structure which can be easily converted and executed on a proposed system or configuration of systems. UNISKEL BM also includes a Live Test Demonstration (LTD) procedure which may be tailored to an agency's requirements. This allows the agency to concentrate efforts on 1) building a model of the future workload and plugging the workload characteristics into an existing benchmark framework (this can be accomplished manually), 2) defining the quality of service desired in new systems (in terms of response times for interactive work and overall batch capacity), and 3) testing the benchmarks to assure that the benchmark represents the desired workload.

UNISKEL BM would not be appropriate for an agency buying a minicomputer or desiring only to measure the speed of a CPU when executing specified instruction mixes. It is appropriate for mainframe competitive procurements where the objective is to procure balanced systems to process a known workload under defined quality of service objectives.

1.3 Scope of Paper

Section 2 describes the UNISKEL BM concept including a brief description of the workload model which provides input data to UNISKEL BM. Section 3.0 describes the UNISKEL BM structure and provides a brief overview of its generation and execution. Appendix A describes the procedure a vendor would follow to conduct a Live Test Demonstration (LTD) using the benchmarks. This example procedure is extracted from NASA/JSC's documentation supporting their acquisition action. Appendix A is not attached to this paper. It will be provided separately to anyone having an interest in that level of detail.

2. UNISKEL BM Concept

2.1 Design Objectives

The UNISKEL BM was designed 1) to be as independent as possible from any vendor's system, 2) to minimize biases for or against any particular computer system architecture, and 3) to minimize a vendor's time and effort in support of the LTD.

A study of the architectures indicated that a benchmark which meets the following criteria meets the design objectives:

- Easily transportable
- Representative of the expected workload as pertains to:
  - Workload characteristics
  - Virtual main storage requirements.
  - "Locality of reference" profile. (Locality of reference pertains to the way a program proceeds through its instructions and data, e.g., sequentially, randomly or some other profile.)

The last criterion, if not representative of the real world, could impact performance depending on whether the system being benchmarked is virtual or non-virtual in its storage management. These criteria are addressed further in the following paragraphs.

2.2 Structural Overview

UNISKEL BM is composed of synthetic programs, data to be processed and descriptions of functions to be accomplished by systems processors/utilities. The

synthetic programs have been designed to be fully transportable to any system having an ANSI/78 FORTRAN compiler. The synthetic program can be made to represent different sizes of batch or interactive tasks. The synthetic tasks are characterized by the number of cycles of CPU kernels executed, the number of I/O accesses to disk and tape, the number of lines printed and the voluntary wait (think) time between interactive activities. The structure of a synthetic task is illustrated in Figure 1.

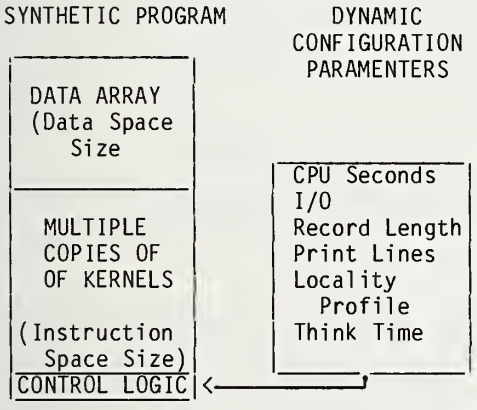


Figure 1. Synthetic Program Structure

The systems processors/utilities functions to be executed are also organized into tasks (by the demonstrating vendor). These tasks utilize the systems processors of the System Under Test (SUT).

These synthetic and system processor tasks are organized into jobs or runstreams. The jobs are further organized into benchmarks, one representing a prime shift workload and one representing a non-prime shift workload. The non-prime shift benchmark is batch only. The prime shift benchmark contains interactive activities and background batch jobs. The prime shift benchmark is designed to be demonstrated utilizing a Remote Terminal Emulator (RTE). An RTE is illustrated in Figure 2.

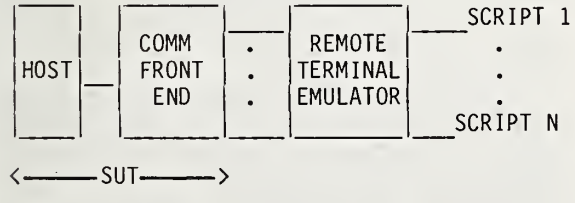


Figure 2. Remote Terminal Emulator

Scripts 1 through N in Figure 2 represent terminal users 1 through N. These emulated terminal inputs cause interactive tasks (synthetic and systems processors) to be executed. These inputs are delivered to the system being demonstrated at rates which are representative of the line speeds of the real communications.

The prime and non-prime shift benchmarks may be generated in various sizes representing different sizes of the representative workload. For example, in NASA/JSC's acquisition specifications, one size of computer system that was acceptable was a system which had the equivalent capacity of two UNIVAC 1108's. Consequently, benchmarks would be executed which represented the prime and non-prime shift workloads of two 1108's under acceptable quality of service objectives. A proposed system of the proper size would have to demonstrate that capability by executing the benchmarks within the threshold elapsed times and response times specified in the specifications.

Different sizes of benchmarks are generated by replicating a base size. For example, the base sizes used by NASA/JSC are shown in Table 1. These sizes were defined as having a Performance Rating of 0.5. The Performance Rating (PR) is a measure of a system's throughput when executing these benchmarks. For example, any system which can complete these PR 0.5 workloads in the specified elapsed time is considered to be at least a PR 0.5 system for the purposes of the acquisition action.

Table 1. Benchmark-Base Sizes PR 0.5

BENCHMARK CHARACTERISTICS	PRIME SHIFT BATCH	PRIME SHIFT INTER-ACTIVE	NON-PRIME SHIFT BATCH
Program Size, 36-bit words.			
Minimum:	10K	10K	10K
Maximum:	76K	45K	125K
I/O Accesses:			
To Disk	29K	90K	22K
To Tape:	19K	0	22K
Jobs:	8	NA	20
Interactive Users:	NA	8	NA
Jobs Using Tape:	1	0	1
Tasks Executed:	94	628	289
Files:	8	80	220
Elapsed Time in Minutes:	<----200---->		210

Different sizes of benchmarks are generated by replicating these PR 0.5 sizes. For example, a system being proposed as having a Performance Rating of 6.0 would demonstrate that capability by executing benchmarks of size PR 6.0. These sizes would be generated by adding 12 copies of the PR 0.5 sizes together. The PR 6.0 size non-prime shift benchmark would consist of 240 batch jobs (20 batch jobs x 12 = 240). The prime shift benchmark would be generated the same way and would consist of 96 interactive users and 96 background batch jobs.

2.3 Scoring and Evaluation

The benchmarks are designed for ease of scoring and of evaluation. Two measurements are used to evaluate performance. These are the elapsed time required to accomplish a well-defined, finite batch or interactive workload (the selected sizes of the benchmarks), and the response times for the emulated interactive users at terminals. For a system being proposed as having a specified performance rating, the benchmark elapsed time, must be equal to, or less than, the elapsed time threshold contained in the specifications. In addition, a specified percentage of all interactive transactions, of the same type, must have a response time equal to, or less than, the response time goals specified.

This method of evaluation is based on the concept of testing a base size of benchmark on a known system and determining the thresholds (elapsed time and response times) of acceptable performance with the representative workload. Based on this, and the knowledge of how much capacity would be required in the known system to meet the requirements new systems are being procured for, a Performance Rating requirement can be determined.

For example, the NASA/JSC benchmarks of size PR 1.0 were tested on JSC's 1108's. The size of the benchmark and/or the elapsed times and response times were adjusted so that the PR 1.0 size could be executed on an 1108 with acceptable quality of service. The future workload to be executed on the new systems was determined to be equivalent to N 1108's. Consequently, systems having a Performance Rating of N are to be procured. A system executing a benchmark of PR N size, within the specified elapsed and response times, has N times the capacity of an 1108 when executing the benchmark.

For example, a multi-processor configuration, or a loosely coupled configuration of more than one system utilizing one input stream for batch, proposed as having a PR7 capability must demonstrate that capability by executing a size PR7 benchmark. A configuration of more than one uncoupled systems can also be accommodated. This is shown in the following example.

LTD REQUIREMENT: TWO UNCOUPLED SYSTEMS HAVING TOTAL CAPACITY = PR 7.0

SCORING EQUATION:

CAPACITY SIZE N + CAPACITY SIZE M .EG. DESIRED CAPACITY

$$\text{CAPACITY (SIZE N OR M)} = \frac{\text{ELAPSED TIME THRESHOLD}}{\text{MEASURED ELAPSED TIME}} \times \text{SIZE}$$

For example: A batch benchmark of sizes PR 3.0 and PR 4.0 were demonstrated on two uncoupled systems to meet a capacity requirement of PR 7.0. Actual elapsed times were 200 minutes for the PR 3.0 LTD and 215 minutes for the PR 4.0 LTD. The elapsed time threshold was specified as 210 minutes. Using the above equations, the actual capacities would be rated as follows:

$$\frac{210 \text{ minutes}}{200 \text{ minutes}} \times 3 + \frac{210 \text{ minutes}}{215 \text{ minutes}} \times 4$$

$$= \text{Combined Capacity}$$

$$3.15 + 3.91 = 7.06 \text{ OK}$$

Size restrictions should be placed on the demonstrator during the LTD. For example, it may be desirable to limit the capacity rating attributed to a system to the size of the benchmark executed. In the example above, a rating of greater than 3.0 would not be allowed because a PR 3.0 benchmark was executed. To receive a 3.15 rating would require the execution of a benchmark of PR 3.5 size. This agency-imposed restriction recognizes that the relationship between the size of the workload and the elapsed time is not linear.

The prime shift benchmark LTD would also include an evaluation of response times. The actual LTD response times would have to meet response time criteria as specified in the statement of work.

Typical response time specifications might be:

1. Of the M different types of transactions, N must meet the following criteria:

a) 85% of executions of a transaction must have a response time of equal to or less than the response time threshold (RTT), plus

b) 98% of executions of a transaction must have a response time of equal to or less than two times the RTT.

2. The balance of transactions (M minus N), must meet the following criteria:

a) 50% of executions of a transaction must have a response time of equal to or less than the response time threshold (RTT), plus

b) 98% of executions of a transaction must have a response time of equal to or less than two times the RTT.

Note that each different type of transaction has its own RTT.

An agency would probably allow no more than 15% of the transaction types to fall into the criteria described in "2." above.

2.4 Workload Representation

UNISKEL BM represents any workload which can be described at the task level in the following terms:

BATCH OR INTERACTIVE TASKS:
- VIRTUAL MEMORY SIZE
- CPU SECONDS TO BE BURNED
- NUMBER OF I/O ACCESSES TO DISK AND TAPE
- NUMBER OF PRINTED LINES OF OUTPUT
- LOCALITY OF REFERENCE-INSTRUCTIONS AND DATA
- THINK TIME FOR INTERACTIVE TASKS

SYSTEMS PROCESSOR/UTILITIES TASKS:
- STATEMENT OF FUNCTIONS TO BE ACCOMPLISHED

These workload characteristics are stated in terms acceptable as input to the UNISKEL BM. Such a workoad model is illustrated in Tables 2 through 4. Table 2 represents an overall workload on one or more systems. Table 3 represents one component of the overall model. Each line item (E&D-1, etc.) will become one synthetic task with the characterstics shown. These individual component models (there is one for each marked entry in Table 2) are normalized in a combined model of individual synthetic or system processor tasks. Several pieces of the prime shift model are shown in Table 4.

Table 2. Composition of CCF Workload

COM-PONENTS	MAJOR PROGRAMS/ PROCESSORS	BATCH	DEMAND	TOTAL
SVDS	SVDS, SVDS1, SVDS2, SVDS3	15%*		11%*
NASTRAN	LINK1, LINK4, LINK 8	1%*		1%*
E & D	ABSLB,TRASYSP, SSFS, ETC.	20%*		15%*
PLOTS	V8 PLOT, V PLOT TRWPLT	5%*		4%*
SYS PROC MISC.	COB,FOR,ASM,ED, OPEN SHOP, OPS, MNGMT.	23%* 36%*	59%* 41%*	32%* 37%*
TOTAL		100%	100%	100%
BATCH/INTERACTIVE (DEMAND) SHARE:		75%	25%	
DISK/DRUM ACCESSES PER CPU+I/O SECS.		16.5	24.2	
TAPE ACCESS PER CPU+I/O SECS.		4.0	3.2	
IO DIVIDED BY CPU		1.38	2.89	

* Model Components

Table 3. E&D Batch Model

PROGRAM	PROGRAM SIZE K WORDS	CPU SUP SEC	I/O ACCESSES DISK	I/O ACCESSES TAPE	FREQ.
E&D-1	63	27	1101		1
E&D-2	63	61	405		3
E&D-3	66	354	2868		1
E&D-4	72	166	3512		2
E&D-5	72	168	14562	3108	1
E&D-6	72	5	492		5
E&D-7	76	385	770		1
TOTALS: Each			30000	3108	14
SUP SECONDS:	1474		818	101	
TOTAL SUP SECONDS = 2393			IO/CPU = 0.62		

Table 4. PR1 Prime Shift
Workload Model

PROG.	PROG. SIZE K WDS	CPU SUP SECONDS	I/O ACCESSES DISK	I/O ACCESSES TAPE	FREQ*THINK TIME SEC
SYSTEMS PROCESSORS - DEMAND:					
ED-1	5	1.456	107		11x28 8
ED-2	5	1.743	129		65x9 8
ED-3	5	.801	73		245x12 8
MISCELLANEOUS DEMAND:					
MISC-1	5	.028	56	2	9x28 1
MISC-2	5	.006	6		9x6 3
MISC-3	5	.020	20		52x20 11
BACKGROUND BATCH:					
SVDS-A	68	213.0	3076		1x0
SVDS-B	75	17.0	1027		1x0
SVDS-C	75	21.0	481		1x0

PRIME SHIFT TOTALS:	CPU	DISK	TAPE	FREQ.
Each:		234428	36968	6180
SUP SECONDS:	3208	6389	1202	
TOTAL SUP SECONDS:	10823	IO/CPU = 2.37		
Disk Accesses/Second = 17.3				
Tape Accesses/Second = 3.7				

* Transactions X Subtransactions

In addition to the workload characteristics shown on the previous tables, the CPU kernels to be used to burn CPU time and the locality of reference profile must be selected. An agency has a choice of using multiple copies of any subroutine or instruction mix (kernels) to burn CPU time.

These kernels are calibrated (CPU milleseconds per cycle and size). "N" cycles of the kernel are executed to burn the desired amount of CPU time. A task (the synthetic program in the benchmark) burning 100 milleseconds of CPU and doing 10 accesses to disk per execution, would do an I/O, burn 10 ms of CPU time by executing, for example, 5 cycles of a 2 ms kernel, do another I/O, burn another 5 cycles, etc. until the 100 ms and the 10 I/O's had been completed.

The kernel selected could be subroutines of applications in the expected workload or could be "canned" instruction or operations mixes. NASA/JSC and the Bureau of Census elected to use primarily subroutines of real applications to enhance representativeness.

The kernels selected to burn CPU time also are involved in the "locality of reference" capability of UNISKEL BM. The size of an activity or task is made up of instructions and data. The instruction size comprises library routines, synthetic program logic, and the multiple kernel copies. The data arrays also contain records which vary in number based on the desired total size of the activity. A synthetic task representing a real task of 20,000 words in size could be generated as approximately 10,000 words of instructions and 10,000 words of data. The instructions might include 2,000 words of overhead instructions and 80 copies of a kernel containing 100 words each (8000 words). The number of copies of the kernel depend on the calibration of the kernel selected and the size to be represented. The number of data records would be selected the same way.

The multiple copies of the kernel and the data records provide two functions. First, they allow the selection of a kernel copy and data record for execution based on a distribution or profile which is similiar to that locality of reference profile shown by the real application. For example, based on a profile entered into the synthetic program, different copies of the kernel and the data records will be selected each time the CPU is to be utilized. Since the synthetic activity is progressing through its instructions and data similiarly to the real application, no bias is introduced for either "paging" or "swapping" memory management systems. The locality of reference mechanism also assures that CPUs with cache memory have realistic "hit ratios."

The major problem presented by this capability of UNISKEL BM is determining what the locality of reference profile is for the real application being represented. A special sampling routine, Code Interruption Analyzer (CIA) was developed to assist in this determination for the NASA/JSC benchmark.

Second, the use of kernel cycles and similiar blocks of data for each I/O, main-

tain the independence from the system upon which the benchmark is developed. Note that the method of accounting for use of resources, seconds, standard units of processing, machine resource units, etc., is immaterial as the work executed is stated in terms of number of kernel cycles for CPU time, number of I/O accesses to various devices, elapsed wall clock time for the total benchmark and response time for interactive activities. This independence is furthered by allowing the LTD vendor to compile the benchmark allowing program sizes to vary with word size and the instruction packing capability of the compiler. For example, an activity within the benchmark may be representing object code, on a UNIVAC system, of 30,000 36-bit words containing instructions and data. This same activity, after being converted and compiled on a CYBER system, could be only 18,000 60-bit words in size. This is acceptable for UNISKEL BM.

2.5 Other Features

UNISKEL BM is a framework within which all types of workload can be represented. For example, data base management systems and transaction processors in the SUT could be exercised by jobs designed to functionally exercise those processors. To date, neither of the users of this framework have had a requirement to exercise such capability.

There is also a framework for meeting floating point accuracy requirements. This is accomplished by the selection of the kernels to be executed. In the NASA/JSC application, some kernels had a requirement for 14 decimal digets of accuracy. This would require some systems to execute those kernels using double precision.

Special kernels could also be designed to exercise special capabilities required in systems to be acquired. For example, kernels could be designed to verify special arithmetic or logical capability of a proposed system.

2.6 UNISKEL Limitations

The primary limitations on UNISKEL BM are two. First, program sizes of less than 10,000 36-bit words cannot be represented due to the overhead associated with the synthetic program. We found this not to be a problem in the NASA/JSC benchmarks.

Second, the generation and testing of new benchmarks utilizing UNISKEL BM, even assuming the workload and locality of reference characteristics are available, represent a significant task both in terms of human resources and computer time. The generation of these benchmarks by a new using agency is the subject of paragraph 3.

It should be noted that although significant effort is required, the human and computer resources required to build benchmarks using UNISKEL BM is probably an order of magnitude less than starting from scratch and building benchmarks utilizing synthetic programs.

3. Benchmark Generation

This section describes how an agency uses the UNISKEL framework to build a benchmark. Each step of the process is discussed briefly; then automatic generation tools are discussed. Figure 3 illustrates the benchmark generation process.

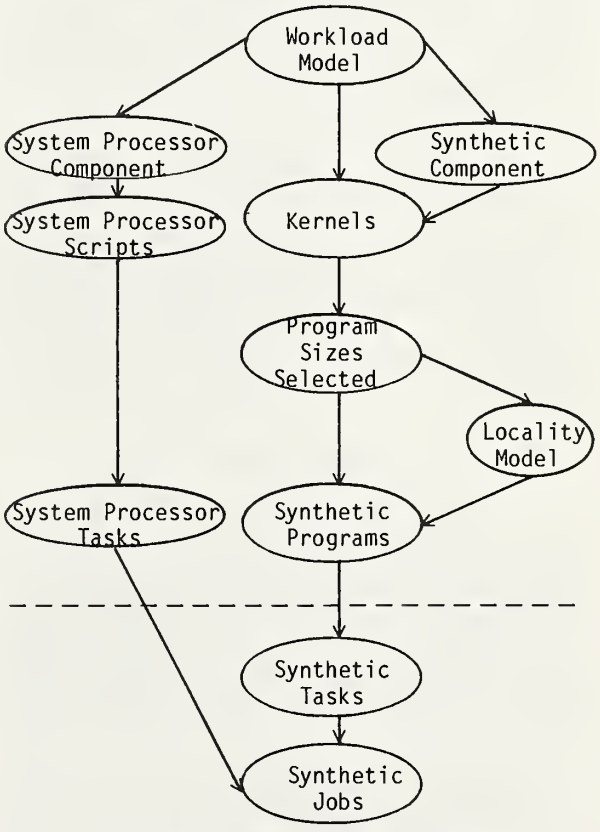


Figure 3. Benchmark Generation

The first step is to determine from the workload model which parts of the benchmark will be represented by synthetic tasks and which by system processor tasks. As

discussed, a system processor task is the execution of a standard system program to accomplish specific work. A synthetic task is the execution of a synthetic program to consume the same resources as a user written program.

In the case of the non-prime shift workload model presented in section 2.4, 23% of the work is accomplished by system processors, and 77% by synthetic tasks. These percentages apply only to the baseline system, as system processor functions may require different resources on different systems. For example, a text editor command requiring 80 Service Units to execute on an IBM system may require 0.1 SUP seconds to execute on a UNIVAC system. This is based on differences in accounting methods, system speeds, operating systems, and software efficiency. There may also be a difference between systems for synthetic tasks due to compiler efficiency, etc. These differences are not undesirable, as the procuring agency's requirements are not for a certain amount of service units to be consumed, but are for a certain amount of work to be accomplished in a specified amount of time.

Utility tasks such as text editing, file copying, and compilation are best represented by system processor tasks. These tasks are more expensive than synthetic tasks for the agency to develop and for the vendor to convert, but more accurately represent the real workload and differences between competing system software. Some utility tasks and all user programs will be represented by synthetic tasks.

3.1 System Processor Tasks

A system processor task is constructed from a functional description (script) of a sequence of operations to be performed. This script, commonly written in English, is translated into JCL for a particular system. If the task is to be interactive, think times are also specified. For example, the script for a text edit system processor task may be:

```
Change all occurrences of ENT to ENTR.
List lines 12 through 50.
On line 50, change 7.0 to 7.5.
On line 48, change RTRY to RETRY.
List the last line of the file.
```

The workload model includes frequency of execution and resources but due to system accounting limitations may omit number of transactions per interactive execution and transaction think time. The agency must estimate any missing factors, then develop by trial and error a script which consumes the proper resources and seems "reasonable" for the real user activity being emulated.

In a competitive procurement these scripts must be designed with conversion in mind. When writing a script for system A, how the same function would be accomplished on system B must be considered. Then the function must be clearly specified in such a way that vendor B will accomplish it in a realistic way, neither taking unfair advantage nor being unfairly prevented from using a special feature of system B. For example, if a vendor has a text editor feature of "change string, and list changed line", the agency should not force him to use two seperate commands "change string", and "list changed line". If a vendor has a single editor command that accomplishes all five steps of the above example, he should not be permitted to use it, since a user would be unlikely to work that way. Command stacking, entering multiple commands on one line, may be permitted, however.

3.2 Synthetic Tasks

Building synthetic tasks requires several steps. First, CPU burning kernels are chosen. A kernel is a subroutine that does a certain amount of work; that is, on the baseline system it burns a calibrated amount of CPU time. Then the synthetic program sizes and composition are determined, and the locality of reference model is incorporated if desired. Then the synthetic programs are compiled and linked. A synthetic task is an execution of a synthetic program with parameters which select CPU time, number of I/O's to disk and tape, disk file size and record length, number of lines of print, and frequency of locale change. The selection of a synthetic program to execute for a task selects the kernel type (and therefore the instruction mix) as well as the memory size. These steps are described in more detail below.

3.2.1 Synthetic Program Structure

Figure 4 shows the overall structure of the synthetic program. For the data array, there is simply a declaration of an array in blank common. Also contributing to the data space size (D-size) of the program are the local data for each kernel and for the control logic.

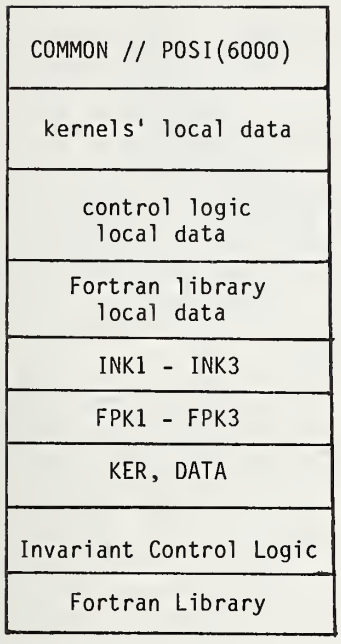


Figure 4. Program Composition

The control logic comprises the main program SYNTH, the input/output routine IOW, and other routines, all of which are invariant whatever the type and size of the program. The control logic also includes two routines KER and DATA which are specific to the program size and type. They have encoded in them knowledge of the number and names of kernels and the length of the data array. The last components of the control logic are the Fortran library subroutines whose size and number is system and compiler dependent.

A program uses one or more types of kernels. There are enough replicas of each kernel type to achieve the desired instruction space size (I-size). For example, a program using the general floating point kernel (FPK) and the integer Bucholz kernel (INK) may include FPK1, FPK2, FPK3, INK1, INK2, and INK3. The only difference between the kernels of a type is the entry point name.

After initialization, the interactive synthetic program sends a THINK command to the RTE. The RTE delays the specified time (THINK time), then sends the time of day. Then the synthetic program executes a transaction and sends another THINK command. The response time is the time between time of day being sent and THINK being received. The batch execution of a program is simply the execution of a single transaction without the RTE dialog. Figure 5 shows the execution flow of a transaction. The CPU time for the transaction is interleaved with the proper number of I/O's and locale changes.

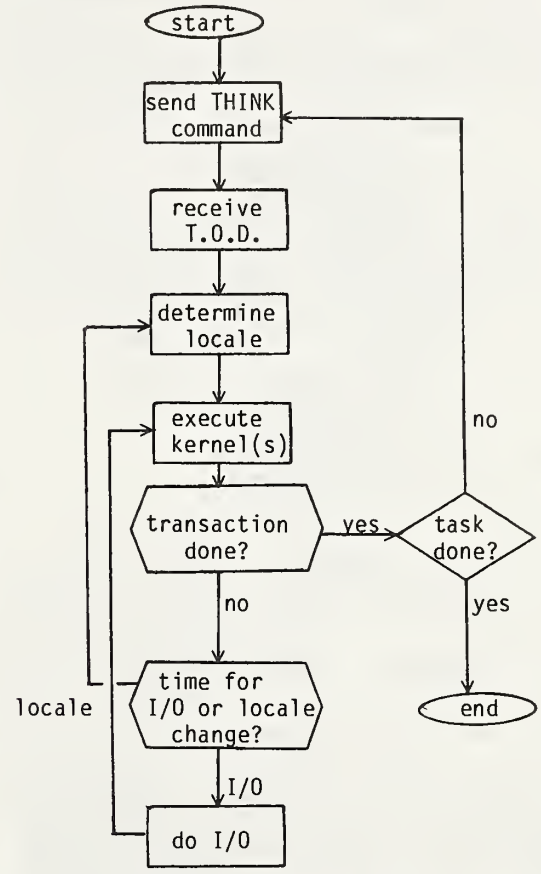


Figure 5. Execute Transaction

Each of these bursts of CPU time is generated by executing one or more kernels (Figure 6). It is determined how much of the burst to consume in each kernel type. For example, suppose it is determined by the Code Interruption Analyzer, or by estimation, that a program uses 60% INK and 40% FPK. A 15 millisecond burst would be divided 9 ms INK and 6 ms FPK. If one pass through INK has been calibrated at 0.9 ms and one pass through FPK at 1.2 ms, the burst will be done by 10 executions of INK and 5 of FPK. These multiple executions may be accomplished by looping around a call to the kernel, or by passing a loop count to the kernel which loops internally. This selection is made to best represent the actual frequency of subroutine calls.

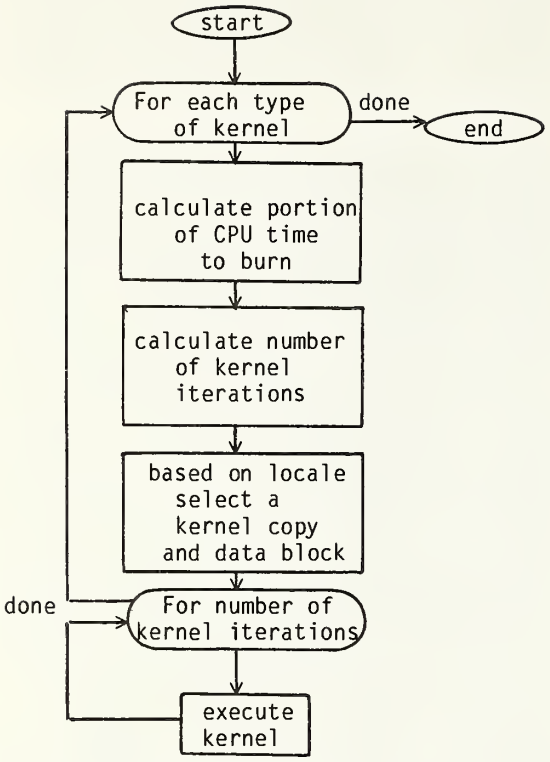


Figure 6. Execute Kernels

3.2.2 Kernels

The kernels used should be representative of the agency's workload because this relatively small portion of the benchmark code consumes the bulk of the CPU time. They could be actual subroutines from the real workload, or could be code which provides the same frequency of logical and arithmetic operations or high level language operations as the real workload. Thus, though the benchmark programs are synthetic, they execute very much like real programs.

The kernels may be written in FORTRAN or in any language callable from FORTRAN. Thus, though UNISKEL is FORTRAN based, it has also been used to represent COBOL workloads. The kernels should conform to applicable ANSI language standards in order to eliminate or minimize the conversion effort required by various vendors. Real subroutines may be modified to conform to the necessary transportability standards while remaining representative of the workload.

Other restrictions may be followed which make benchmark generation easier. The compiled size of a kernel should be small enough that the task sizes can be represented with the desired granularity and large enough to allow the largest desired program to be constructed. The size of the instruction space of a program is accurate to within the size of one kernel. But compiler and UNISKEL limitations restrict the maximum number of kernels which may be included in a program. So in order to represent large programs, large kernels may be required.

Similarly, the CPU time consumed by one pass of a kernel should be low enough that the desired task CPU times can be achieved with the desired granularity. The CPU time consumed by a synthetic task can be accurate only to within the CPU time of one kernel execution. But the overhead of the control program is proportionally higher for short kernel execution times. This does not mean that the task CPU time will be wrong, but that task calibration will be more difficult, and that the instruction mix will not be so representative. In order to avoid this, longer executing kernels may be required.

The execution of a kernel should be independent of other routines. A kernel should have no output parameters and should not modify its input parameters. Local variables should be minimized and common variables should be used with caution. Particularly to be avoided is non-uniform execution time and run-time data initialization. A kernel reads its parameters, and works in blank common, thus exercising the locale mechanism.

The CPU time of the kernels must be calibrated to within the required precision. At NASA/JSC the method used was a FORTRAN main program that queries the CPU time used, executes a kernel a large number of times, and again queries the CPU time used. The CPU time of the kernel is the total CPU time divided by the number of times executed.

3.2.3 Program Sizes

The compiled size of each the components of a synthetic program must be determined. Then the number of kernel copies and amount of blank common necessary to achieve the desired instruction and data sizes on the baseline system are calculated.

3.2.4 Locality of Reference

The locality of reference may be specified for a synthetic program by means of a reference histogram; this is called directed locale. Blank common is logically divided into a number of blocks; for JSC we arbitrarily chose 1000 word blocks. A locale is achieved by executing a certain kernel copy and referencing a certain block of data. For each kernel copy and for each data block, a probability of access is entered in the histogram. When the synthetic program makes an instruction locale change, it uses a pseudo- random number generator to choose a kernel copy on the basis of the instruction histogram. When it makes a data locale change, it uses a pseudo- random number generator to choose a data block on the basis of the data histogram. It is not necessary that the sequences of random numbers be the same for different vendors. The agency may permit the vendor to use a convenient modulus, as long as the algorithm remains intact.

These histograms are compiled into the synthetic programs. Two locale parameters supplied as parameters at run time are the I-locale threshhold and the D-locale threshhold. The former is the number of CPU milliseconds to consume between instruction locale changes. The latter is the number of instruction locale changes to make between data locale changes.

If a locale model is not available, random locale may be used, in which case the probability of choosing each locale is equal. The I-locale and D-locale parameters are entered at run time as with directed locale. The working set of a program on a virtual system may tend to be larger with random locale than with directed locale for any given locale threshholds. So if random locale is to be used, slightly longer locale threshholds may be chosen to compensate.

Figure 7 shows a locale histogram for a program of 18 kernels and 19 blocks of blank common. The histogram value represents the probability of access of a locality. The peaks in the histogram correspond to high activity subroutines, loops, and data structures in the real program. Note that there are 6 kernels and 10 data blocks which are rarely referenced. On a virtual memory system these might not be in the program's working set; but they would be referenced occasionally. Random locale corresponds to a flat histogram: no locale is subject to high activity or low activity.

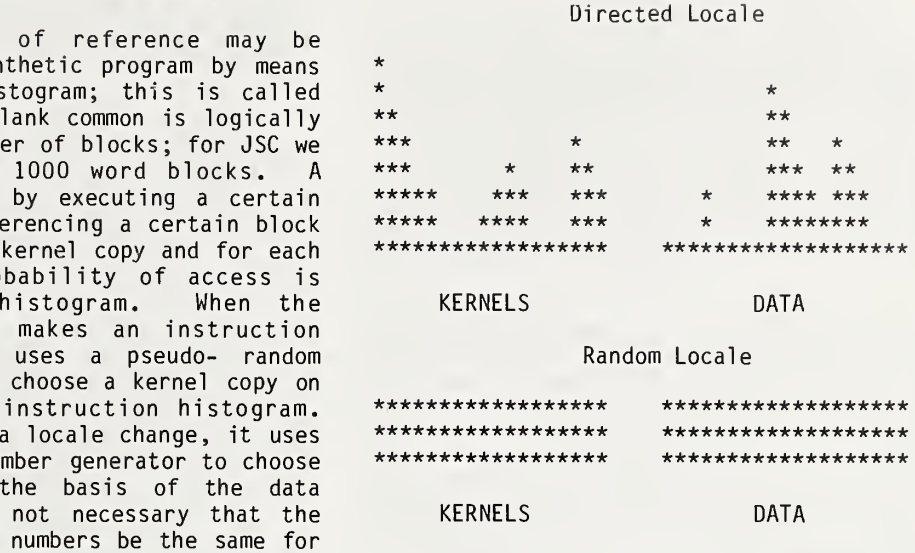


Figure 7. Directed and Random Locale

3.3 Benchmark Jobs

The number of jobs in which to apportion the tasks of the workload is first decided, then the jobs are created. The minimum number of jobs is the degree of concurrency needed in competing systems. The goal is to allow all vendors to use their machines to full potential. Having more jobs also minmizes tailoff, the time at the end of the benchmark during which the system is not fully utilized due to insufficient concurrency. The maximum tailoff is on the order of the execution time of one job. The factors limiting the number of jobs are granularity of task execution times and overhead of job initiation and termination.

A job, consists of JCL for initiation (job identification, file assignment, etc.), JCL for each task, and JCL for termination. Thus, for example, job 1 may consist of tasks 3, 7, 7, 2, 14, 3, and 10. Job 2 may consist of tasks 7, 2, 8, 17, and 3. The jobs are constructed to have roughly equal execution times to minimize tailoff. The executions of tasks within a job are independent, and so may be in any order.

Support jobs are also created which catalog, initialize, and delete the benchmark files. These run before and after the benchmark proper. In the NASA benchmark, source files are initialized by copying from base source files. Binary files are initialized with date and timestamp records.

3.4 Batch and Interactive Benchmarks

The work required to build a benchmark down to the level of job generation (dotted line in Figure 3) should be significantly less than that required by conventional methods. Even greater savings can be realized if more than one benchmark need be generated. This will frequently be the case as an agency's requirements evolve during the course of benchmark specification and construction.

From this job generation level the benchmark implementor may change: PR size of benchmark, number of jobs, number of tasks, and interactive think times. And for synthetic tasks the implementor may also change: CPU, I/O, lines of print, output record length, output file size, and frequency of locale change. These possibly extensive modifications can be made and a new benchmark generated and calibrated in a matter of days. Even if a complete benchmark generation is done, this will be easier the second time after the agency is familiar with the techniques. The generation of a benchmark can be simplified by the use of automatic generation tools, described below.

3.5 Generation Tools

The UNISKEL approach can be used to manually generate a benchmark. Or, if access to a UNIVAC 1100 system can be arranged, there are a number of automatic generation routines available, written in UNIVAC Symbolic Stream Generator language. This section briefly describes these routines.

One routine (SKEL/PROGRAMS) determines the composition of the synthetic programs based on the desired kernel(s) and program size, the control program size, the kernel sizes, the FORTRAN library size, etc. It considers both the instruction space size and the data space size. It detects certain errors which would make the desired program size infeasible.

Based on the above criteria, SKEL/PROGRAMS generates the unique source code for each program. It also generates routines which compile the source code and link it into executable programs. It allows different (FORTRAN callable) languages to be used for the kernels if desired.

Another routine (SKEL/ASSIGN) determines the tasks which comprise each job based on estimated task elapsed time. All jobs are of approximately equal length to minimize tailoff. A related routine (SKEL/ORDER) randomly reorders the tasks within each job.

Based on the job composition, SKEL/RUNBUILD generates the JCL or RTE scripts for each job. It considers the files neccesary for each run and generates the JCL to assign them. It also generates support routines which catalog, initialize, and delete benchmark files.

Other routines assist in calibration and documentation by producing formatted tables describing the tasks and jobs, and by producing a job which executes each task once. Additional information on these routines will be furnished on request.

4. Benchmark Execution

4.1 Benchmark Testing

It is necessary to consider the execution of the benchmark by the agency and by the vendors. The agency will run the benchmark for pre-release verification, for accurate sizing of its own system, or for system tuning. The vendor will convert and generate a benchmark and will conduct a Live Test Demonstration to demonstrate performance to the agency.

In the construction of a benchmark, an agency should consider the requirements for testing the benchmark on its own system. For realism, the benchmark typically includes many files, large and small, infrequently referenced as well as frequently referenced, both catalogued and temporary. It will, therefore, require large amounts of mass storage and require special procedures for execution on a general purpose system. This could require that a system's files be "unloaded" prior to testing the benchmark. This is time-consuming and expensive. As an alternative for testing purposes, a second benchmark can be constructed which accurately represents the workload characteristics but which requires less storage. Since it would be almost identical to the real benchmark, it would require very little additional effort but would minimize the cost of testing.

4.2 Benchmark Conversion

Two major goals of benchmarking are often difficult to achieve together. The benchmark should be representative so that a system which performs acceptably on the benchmark will perform acceptably on the

real workload. But it should not be too difficult for vendors to implement, because the vendor's implementation cost may be passed on to the government in terms of higher bids.

Most of the synthetic program source code is a subset of ANSI 66 and of ANSI 78 FORTRAN for easy transportability. Only the I/O routine and the clock routine are written in assembler language and must be recoded by the vendor. These two routines total 50 lines.

Some incompatibilities may remain, however, between compilers said to conform to the ANSI standard. So the modular construction of the benchmark minimizes the amount of code to be converted. The FORTRAN control software modules total 500 lines. For each program the two unique modules (KER and DATA) average 200 lines together. For as many copies as are needed by the largest program there are kernel modules of about 50 to 100 lines each.

All the KER and all the DATA modules are very much alike. All the kernel copies are identical but for entry point name. There are approximately 300,000 lines of code in the JSC benchmark; however, there are only about 1000 lines in the unique modules.

The components to be written or converted are the source code discussed above, the system processor scripts, the synthetic task JCL, the files, and the RTE commands. The system processor scripts should be chosen with conversion in mind. Benchmark files are either text files, binary data files whose structure is dictated by the vendor supplied I/O routine, or files created by system processors, so file conversion should be minimized.

After the system processor task JCL is written, the JCL conversion should be straightforward. Each synthetic task is the invocation of a synthetic program with two lines of parameters. Each synthetic program uses the same file(s) in the same way. Files are neither created nor destroyed except possibly by system processor tasks. Temporary files are assigned at job start and are not freed until job end. Therefore, the execution of one task is independent of the execution of another, and without knowing the containing jobs, the vendor may determine the JCL for each synthetic task.

The JCL for each job in the base workload is constructed. Each job is specified as a sequence of tasks whose JCL can be combined with job initiation and job termination JCL to form the complete job JCL. The rest of the jobs are constructed from the base jobs by changing job names and possibly changing file names.

The vendor converts and compiles the synthetic programs, converts the system processor tasks, generates and replicates the files, and builds the jobstreams. The benchmark is then ready to run.

The main elements of a description of the benchmark for the vendor are descriptions of programs, files, tasks, and jobs. The vendor is supplied with source code for each routine in the benchmark and specifications for each assembler routine. This source code includes one copy of the invariant portion of the control logic, one copy for each program of the variant portion of the control logic, KER and DATA, and many copies of each kernel. The programs are then described by their constituent modules. For example:

```
Program 5:
     SYNTH,SETTR,BMTIME,FFGTST,ICV,IOW,
       LIBRY,LOCD,LOCI,ZOR
     KER5,DATA5
     INK1,INK2,INK3,INK4
     FPK1,FPK2,FPK3,FPK4
```

The benchmark files are described as follows:

```
File 1:
     One copy per job
     temporary
     300KB
     indexed sequential access
     binary data
     initialized by supplied program
```

The tasks are described as follows:

```
Task 1:
     Execute synthetic program 3 with
      parameters
     0,20,448,256,5,2,0,218,5
     5,16,0,1,9
     Disk output to file 9, FORTRAN Unit 9

Task 3:
     Copy file 6 to file 7
```

The jobs are described as follows:

```
Job 1:
     Task 3, Task 7, Task 7, Task 2,
       Task 14, Task 3, Task 10
```

4.3 Verification

The procuring agency will verify at LTD time that the benchmark work was done, that it was done in the required time, and that performance goals were met. UNISKEL provides convenient ways for the agency to do as much verification as it deems necessary. The system clock measures the benchmark elapsed time. The system log verifies that the work was done. The vendor supplied RTE log analysis program verifies that response times were acceptable. These primary checks are easy to use, but may not be reliable enough for the agency's needs, so other spot checks are typically used.

A stopwatch verifies the benchmark elapsed time, as well as correct system clock operation. Print output from some of the jobs is examined to see that the start and stop times from the system log are correct, and that the proper tasks were executed. The timestamped RTE log is examined for the same reason, and to verify response times. The synthetic program writes timestamps in all its output, which are checked for consistency.

In interactive mode, the synthetic program reads timestamps from the RTE, which it prints along with its own timestamps. Prior to LTD, the agency test team may have the system clocks of the RTE and of the System Under Test set to different dates and times. This difference is checked in the RTE log.

The binary files in the benchmark are created by an initialization program that fills them with timestamps. The synthetic program writes over some or all of these files with timestamps of its own. Prior to LTD, the test team selects certain temporary files to be made permanent. The timestamps in these files, as well as those in some other permanent files, are examined after LTD for consistency.

The synthetic program keeps a count of the number of times the kernels are called, and this value is printed for verification. A more rigorous check can be accomplished with an addition to the kernel. The kernels may compute pseudo- random numbers based on a seed supplied by the agency prior to LTD. The final numbers generated by each program execution depend on the seed and on the number of calls to the random number routine. These values may be printed and compared against the known values.

COMPARING USER RESPONSE TIMES ON PAGED AND SWAPPED UNIX BY THE TERMINAL PROBE METHOD

Luis Felipe Cabrera

Department of Mathematics
and
Electronics Research Laboratory
University of California, Berkeley
Berkeley, CA 94720

Jehan-François Pâris

Department of Computer Sciences
Purdue University
W. Lafayette, IN 47907

In this paper we present a comparison of user response times on paged and swapped versions of the operating system UNIX for the DEC VAX 11/780. The technique used was to construct a script that periodically evaluated the system's work-load and measured the system's response times to a set of benchmark programs.

These measurements, collected at two different sites, show that differences of responsiveness observed between the two systems depended much more on the work-loads and the configurations than on the operating systems themselves.

Since we only used standard UNIX tools to build our scripts, they are highly portable and can be installed on any standard UNIX system in a matter of minutes without bringing the system down.

Keywords: terminal-probe method; UNIX.

1. Introduction

A change from one version of an operating system to another one always constitutes a difficult problem for the management of a computer installation. The problems of updating the software while attempting to keep the user's interfaces as unchanged as possible are well known. We will address here another problem, namely, the performance implications of the change.

Numerous tools already exist with the purpose of predicting or measuring the performance of a computer system. In contrast to many of these tools, the method we present here requires no special hardware and very little effort for modeling the system or estimating its work-load. The technique used, which applies to all interactive systems, essentially consists of constructing a script that periodically evaluates the system's work-

load and measures the system's response times to a set of benchmark programs. As we will see, the only drawback of the method is that systems with highly variable work loads will require longer data gathering periods.

Since standard UNIX tools were used to build these scripts, they are highly portable and can be installed on any standard UNIX system in a matter of minutes without bringing the system down. Moreover, the philosophy of the method is by no means specific to UNIX systems and could be applied to other interactive systems.

Section 2 of this paper presents those features of the UNIX operating system which are relevant to our problem. In Section 3 we introduce the data gathering method. In Section 4 we present and analyze diagrams which compare the performance of the installations under the different versions of the operating system. Section 5 discusses some general issues about comparing distinct operating systems. Section 6 has our conclusions.

2. The UNIX System

UNIX is a trademark for a family of time-sharing operating systems developed at Bell Laboratories during the last twelve years [10, 9]. The first version of UNIX was implemented in 1969 on a PDP-7. Since then, several versions of UNIX have been developed at Bell Laboratories. They were primarily aimed at various members of the PDP-11 family of minicomputers and, in 1978, Bell Laboratories released a version of UNIX aimed at the new VAX 11/780. Despite the fact that the VAX hardware was designed to support a paged virtual memory system (DEC's own VMS), the successive versions of UNIX developed at Bell Laboratories did not support paging. This motivated a group at the University of California, Berkeley, to implement a virtual memory extension to UNIX for the VAX [1]. This extension used a Global LRU (Least Recently Used) page replacement strategy with use bits simulated by software and is now an integral part of what is known as the "Berkeley UNIX."

One of the most remarkable features of UNIX is its user interface. On logging in, each user is assigned a special process containing a command interpreter that listens to the terminal. This command interpreter, known as the shell [2], parses the input line decoding the command requested, its flags, and the arguments passed to it. The shell then exec 's the command. The shell also has a redirection mechanism that allows it to read command lines stored in files. Users can thus define sequences of shell commands, known as shell scripts, and store them in files awaiting later invocation. The versatility of these scripts is greatly enhanced by the fact that the shell language also contains control-flow primitives, string-valued variables and even simple arithmetic facilities. Thus, building a sequence of commands that will be repeatedly executed at fixed time intervals is on UNIX a nearly trivial task.

The portability of our tools was also facilitated by the fact that UNIX handles automatically all file allocation decisions. UNIX files are accessed through a hierarchy of directories and the typical user is not aware of the physical location of the files he or she manipulates.

3. The Data Gathering Method

Our strategy for monitoring each system's responsiveness was the same one used in Cabrera's performance analysis study of UNIX [3]. It consists of running periodically a set of predefined benchmarks in a totally automatic way. This was achieved by writing a shell script that is essentially a loop containing the benchmarks together with commands that gather statistics about the work loads and measure the time it takes each benchmark to complete. Each time the script has cycled through the execution of these commands, it executes a sleep command that suspends its execution and then wakes it up after a predetermined number of seconds.

This script is then run as a background job (with the same priority as any user process) during the operation time of the system.

This data gathering method can be categorized as a time-sampling method [7] and in fact is very similar to Karush's terminal probe method [8]. By using it, we measure the work load of the system as well as the dependency of our performance indexes on the underlying equipment. We are thus evaluating the performance of an installation.

Although running a script affects the load of the system--and thus its responsiveness, it was felt that this would not affect the validity of our comparison study since each system would be presented with the same script. The main purpose of our comparison experiment was precisely to observe how each system reacted to this stimulus.

Our commitment to use only standard UNIX tools decided us upon the usage of the time command for measuring the completion time of each benchmark. The time command returns, upon completion of the command it prefixes, three measurements; response time, system time and user time. Response time is only accurate to the second (time truncates, does not round off). This low resolution of time , together with our desire that no individual measurement be off by more than 10%, led us to restrict ourselves to benchmarks that would never take less than five seconds to complete.

4. The Measurements

The design of a comparison study like ours is faced with many constraints and trade-offs. There are basically three main areas in which major decisions have to be made: the selection of the benchmarks, the length of the data gathering period and the representation of the data.

Since our method measures the performance of an installation, i.e., the work load is also included in the observations, a desirable goal is to use as benchmark a task representative of the user's tasks. At the same time, one must try to capture the work load through some characterization. Then, the response time of the task is plotted against the characterization of load.

We have chosen tasks which exercise the system in a "natural" way (i.e., we did not run tasks which would a priori perform better in a paged environment or in a swapping environment) and which were representative of the user's activities at the time. Since text processing constituted on both sites a significant part of the system's load, we decided to use, in both sites, the command man man as benchmark The man command retrieves and formats the entry for any given command out of the on-line copy of the UNIX Programmer's manual; thus man man retrieves and formats the manual entry for the man command. This task is interesting because the entry is retrieved from disk using the widely used formatting program nroff. To avoid screen output problems when running the script, the output of man man was sent to /dev/null instead of sending it to a real terminal. This had the effect of discarding the already formated text of the retrieved page. The length of each data gathering period was determined at each site and will be discussed in the next two subsections. As for the display of the data, after an exploratory analysis of our measurements it was decided to use 75-percentile curves. In Figures 1 and 2 we display four curves for the task man man using the number of logged in users as characterization of load. The four curves are the median (50-percentile), mean, 75-percentile and 90-percentile of the distribution of response times per value of our work load characterization.

Figure 1 displays these curves for the swapping system and Figure 2 for the paging system. In them we may appreciate how the influence of the outliers becomes apparent in the 90-percentile curve. We may also notice that the median of the response time samples are con-

sistently lower than the mean of the samples. This happens in any distribution which is skewed towards the lower range values or, whose outliers lie in the higher range of the values. These two characteristics were common to all our data. To deemphasize the effect of outliers we chose to use the 75-percentile curve to display our data. This decision was partially motivated by unavailability of abundant data from one of the sites; 95-percentile curves would have been too much influenced by outliers.

We decided on two single-variable characterizations of load: number of users logged in and number of user processes. This latter index represents the number of processes which have been generated by user commands. It does not include any system generated processes.

4.1. The Berkeley Measurements

This analysis was based in 431 data points for the swapping system and 1455 points for the paging system generated during the second half of 1979. Throughout the data gathering period the system's hardware remained unaltered. The work load, however, underwent a substantial change (by very large programs such as the VAX version of the algebraic manipulation system MACSYMA [6]) to thoroughly exercise the memory management capabilities of the new kernel.

Given the small amount of main memory the system had at the time, these large address space programs did influence significantly the work load of the system. The hardware configuration of the system was a VAX 11/780 CPU, 512K bytes of main memory, 16 ports, two RP06 disk drives with a DEC disk controller and one TE 16 tape drive.

We shall present data for three tasks: the command man man , the execution of a cpu-bound job and the compilation of a short C program. The choice of the cpu-bound job was motivated by our desire to observe the effect paging had on a task that would almost never be swapped out: its size is 2K bytes.

Figures 3 and 4 display the 75-percentile curves of the response time for the command man man. On both figures we see that the swapping system performs slightly better than the paging system at higher load levels. This difference can be justified by the dramatic change in the workload which occurred when the paging system was brought up. Since we do not have data from stand-alone measurements, we cannot rate the two systems solely on these data, but it becomes apparent that neither outperforms the other one throughout all the values of load.

Figures 5 and 6 display our data for the CPU-bound job. This job consists of two nested loops and an inner block of statements. A 9 statement sequence of integer arithmetic operations is executed 100,000 times. The size of the object code is only 2K bytes--i.e. four 512 byte pages-- and thus its probability of being swapped out is quite small. It is quite interesting to observe that the paging system outperforms the swapping system under both characterizations of the workload. This can be explained in terms of the additional I/O activity existing in the paging system. Indeed, since our job is very small in size and performs no I/O, the only impediments that may block it to run to completion are time-quantum expirations and multiprogramming delays. Thus our cpu-bound job is either running or in the ready queue. In the paging system it more often gets the CPU because of other jobs being blocked by page faults. This fact makes us believe that trivial tasks should perform better in the paged version of UNIX.

Figures 7 and 8 display our data for the C compilation of a small C program. This is the only observed task where the swapping system appears superior. This is specially clear in Figure 8 where the number of user processes characterization of load is used. We believe that the difference in performance observed in this case is mostly due to the change in workload which occurred in the system. The

execution of processes whose size was much larger than the available memory created high contention for memory. Thus medium size processes like the C compiler would always be losing their used pages and paging in those pieces of code needed for further processing. Their resident set would never be allowed to remain at any reasonable size. The effect of large processes was quite noticeable at the time, specially when processes like VAXIMA [6] would do garbage collection through a 2 Megabyte address space. At those points, response time for all commands would degrade ostensibly.

4.2. The Purdue Measurements

These experiments were performed during the Summer of 1980 on the VAX 11/780 of the Department of Computer Sciences at Purdue University when the Department decided to switch from UNIX version 7 to Berkeley UNIX. At that time, the machine had 3 Megabytes of main memory, 56 ports, three RM03 disk drives on Massbus 0 and one TE16 tape drive on Massbus 1.

We chose to run as benchmarks the UNIX command man man , which formats and displays the entry of the UNIX manual describing the command itself as well as a small script containing "man man" and several small tasks. As said before, this choice was motivated by the fact that text processing constituted then the application concerning the greatest number of users. In UNIX version 7, the man command is implemented as a shell script while Berkeley UNIX uses directly executable code. Since this latter alternative is inherently much faster than a script, which must be interpreted by the shell at each execution, a direct comparison of the response times for the man man command would have been grossly unfair to the version 7. We thus decided to run the version 7 script in place of the original man command in our measurements with the Berkeley UNIX. As a result, the response times for the man man command measured at Purdue were much higher than those observed at Berkeley, where the same problem did not occur.

Because of the short interval of time left between our decision of running the script and the scheduled switch from swapped to paged UNIX, we were only able to collect 152 measurements with the swapped version of UNIX. This left us with about ten observations for each load level, as expressed either by the number of users logged in or the total number of user processes. These values were obviously much lower than those required to obtain acceptable estimators of the 75 percentiles of the response time.

Faced with the same problem, but on a much smaller scale, one of the authors [3, 4] decided to cluster neighboring load levels with insufficient numbers of observations. For instance, if 30 was the minimum acceptable sample size and there were 19 observations corresponding to 15 users logged in and 12 observations corresponding to 16 users, the two sets of observations would be merged into a single set of 31 observations and this set made to correspond to a work load of $(15+16)/2 = 15.5$ users. The same approach applied to our Purdue data would unfortunately resulted into too little load levels after clustering. We decided therefore to use a scheme in which the 75 percentile for the load level i would be computed taking in account all the measurements at load levels i-1, i and i+1. This filtering greatly reduced the influence of outliers on the 75 percentiles. It has, however, the unfortunate side-effect of introducing a positive correlation between neighboring values on the percentile curves and therefore should not be considered as a substitute for more measurements

Figure 9 and 10 display the 75-percentile curves of the response time for the script version of the man man command. As one can see on both figures, the response times for the swapping version of UNIX appear to be somewhat higher than those corresponding to the paging UNIX and exhibit also a more erratic behavior. These results apparently do not agree with those observed on

the Berkeley VAX. One should however point out that the Purdue VAX had a much larger memory and that all our measurements relative to the paging version of UNIX were made during the month immediately following the conversion to Berkeley UNIX, thus before any change in the working habits of the users could have occurred.

Another point to mention is that the curves representing the 75-percentiles of the response time against the number of user processes are somewhat better behaved than those corresponding to the number of users. This suggests that the number of user processes is a better estimator of the system's workload.

5. Discussion

One can conclude from our measurements that the switch from UNIX version 7 to Berkeley UNIX did not alter significantly user's response times on any of the two monitored installations. This conclusion, however, depends on the work loads observed on the two machines and is by no means an answer to the question: "Which one of the two operating systems is better?"

Paging was introduced, more than twenty years ago, in order to allow bigger jobs to run on machines then characterized by very small main memories. It became later a nearly universal feature of large-scale multi-access systems because it allowed to keep more jobs simultaneously residing in main memory, thus avoiding swapping delays. As it was quickly noticed, paging unfortunately never comes for free: it requires special hardware, introduces a fair amount of software overhead and performs very poorly with programs exhibiting scattered reference patterns.

One may then question the efficiency of paging at a time where main memory is so cheap that it becomes possible to keep residing in memory enough conversational users to saturate the CPU of a machine like the VAX. Would this be the normal case, jobs would typically run without having been ever swapped out during their execution. This would remove any incentive for implementing a paging scheme. Moreover, it would even make straight swapping more effective than paging since it is usually more efficient to bring into memory the whole address space of a program in a single I/O operation than by successively fetching faulting pages.

This argument does not, however, stand against the well-known fact that larger address spaces have always resulted in larger programs. In fact, one of the strongest motivations of the Berkeley UNIX was to provide the larger address space required by algebraic manipulation programs. Our measurements on the Berkeley VAX show indeed that the switch from swapping to paging has significantly altered the system's work load by allowing bigger jobs to run on the machine. Although it did not appear in our data, the same phenomenon also occurred on the Purdue VAX, whose current work load is strikingly different from the one existing before the switch to Berkeley UNIX.

The main advantage of one operating system over the other one is thus more a question of increased capabilities than faster response times. As a cynical observer could point out, the switch to a better system might be accompanied by an increase of the average response time resulting from the increased demands placed on the system's hardware.

6. Conclusions

We have presented here a simple method for comparing user response times on two versions of an operating system. The method used consists of constructing a script that periodically evaluates the system's load and measures the system's response times to a set of benchmark programs. This method can be very easily implemented on UNIX or any other system that has features allowing to submit periodically a set of tasks and to collect information on response times and system's workload. It does not require the system to be brought down and does

not affect the normal operation of the installation.

Results concerning a paged and a swapped versions of the VAX 11/780 UNIX system show that the observed differences of responsiveness between the systems depended more on the workloads and the configurations than on the operating systems themselves.

Thus our methods should not be construed as a technique for comparing the inherent merits of two operating systems, but rather as a tool giving prompt quantitative answers on the responsiveness of any particular installation.

Acknowledgements

The work reported here was supported in part by the NSF grant MCS 80-12900. The authors express their gratitude to their respective departments for having provided the facilities used in this study.

References

[1] Babaoglu, O., W. Joy and J. Porcar, Design and Implementation of the Berkeley Virtual Memory Extension to the UNIX Operating System, Department of EECS--Computer Science Division, University of California, Berkeley, (1979).

[2] Bourne, S. R., The UNIX Shell, The Bell System Technical J. 57, 6 Part 2 (Jul.-Aug. 1978), 1971-1990.

[3] Cabrera, L. F., A Performance Analysis Study of UNIX, Proceedings of the 16th Meeting of the CPEUG, Orlando, FL, October 1980, pp. 233-243.

[4] Cabrera, L. F., Benchmarking UNIX : A Comparative Study, in Experimental Computer Performance Evaluation (D. Ferrari and M. Spadoni eds.) North-Holland, Amsterdam, Netherlands,pp 205-215.

[5] Digital Equipment Corporation, VAX 11/780 Technical Summary. Maynard, Mass., 1978.

[6] Fateman, R. J., Addendum to the Mathlab/MIT MACSYMA Reference Manual for VAX/UNIX VAX-IMA, Department of EECS--Computer Science Division, University of California, Berkeley (Dec. 1979).

[7] Ferrari, D. Computer Systems Performance Evaluation, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[8] Karush, A. D., The Benchmarking Method Applied to Time-Sharing Systems, Rept. SP-3347, System's Development Corporation, Santa Monica, CA, August 1969.

[9] Kernighan, B. W. and J. R. Mashey, The UNIX Programming Environment, Computer 14, 4 (Apr. 1981), 12-24.

[10] Ritchie, D. M. and K. L. Thompson, The UNIX Time-Sharing System, Comm. ACM 17, 7 (Jul. 1974), 365-375. A revised version appeared in The Bell System Technical J. 57, 6 Part 2 (Jul.-Aug. 1978), 1295-1990.

[11] Ritchie, D. M., S. C. Johnson, M.E. Lesk and B. W. Kernighan, The C Programming Language, The Bell System Technical J. 57, 6 Part 2 (Jul.-Aug. 1978), 1991-2019.

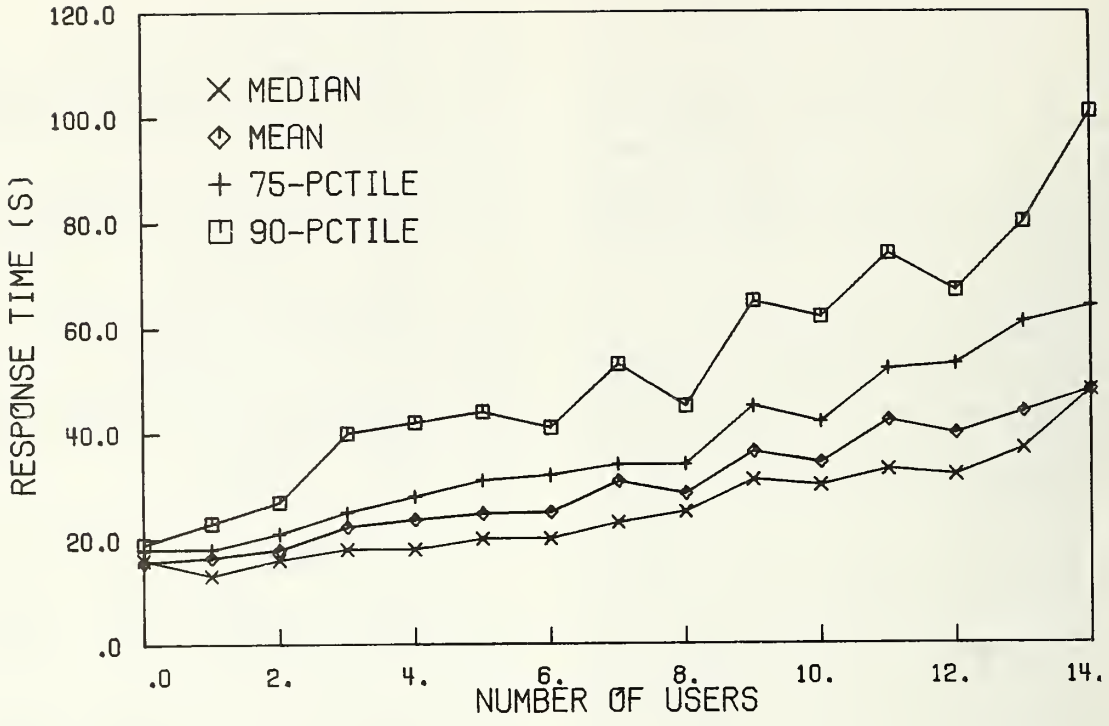


Figure 1 Median, Mean, 75 and 90 percentiles for ""man man"" and Swapped UNIX

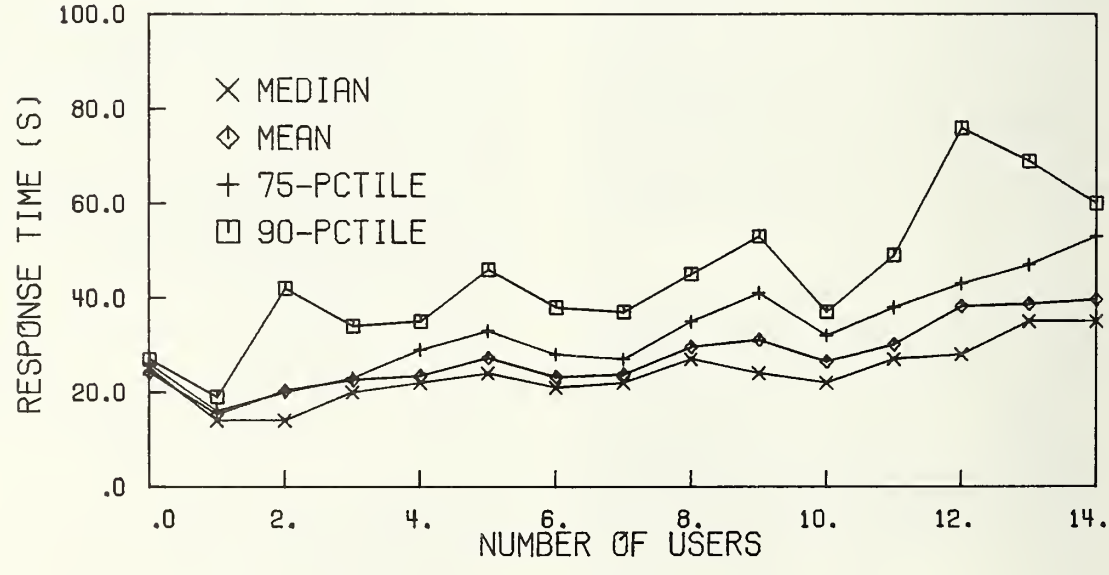


Figure 2 Median, Mean, 75 and 90 percentiles for "man man" and Paged UNIX

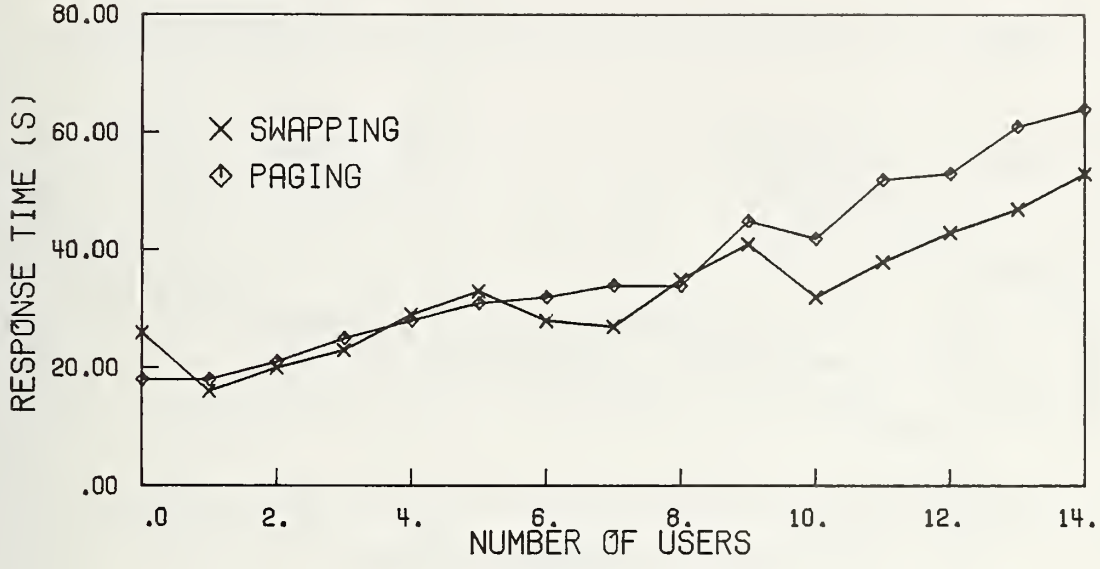


Figure 3 Response Time 75-percentile vs Number of Users for the "man man" command

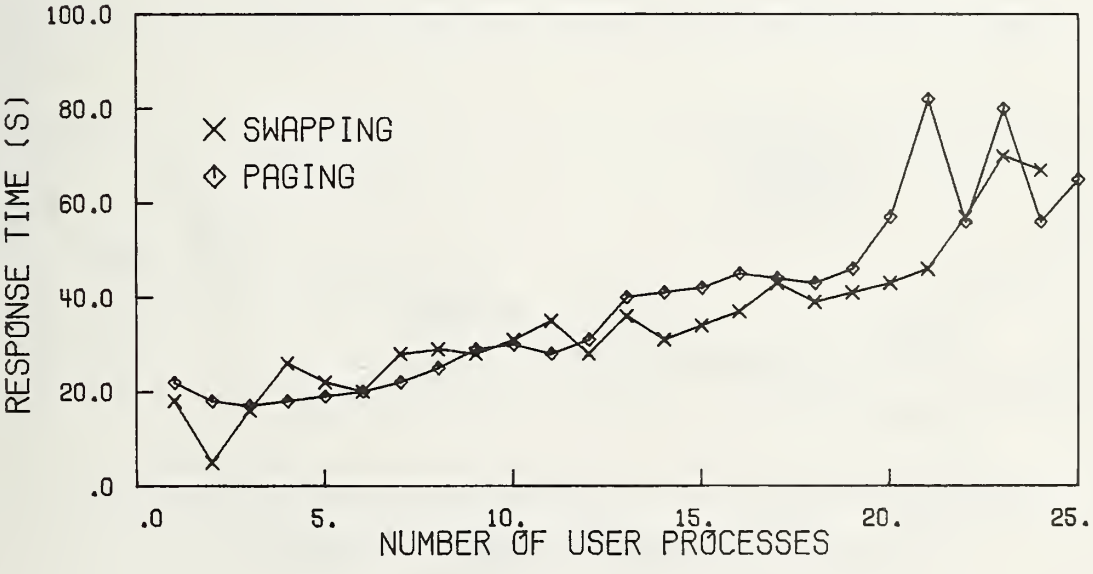


Figure 4 Response Time 75-percentile vs Number of User Processes for the "man man" command

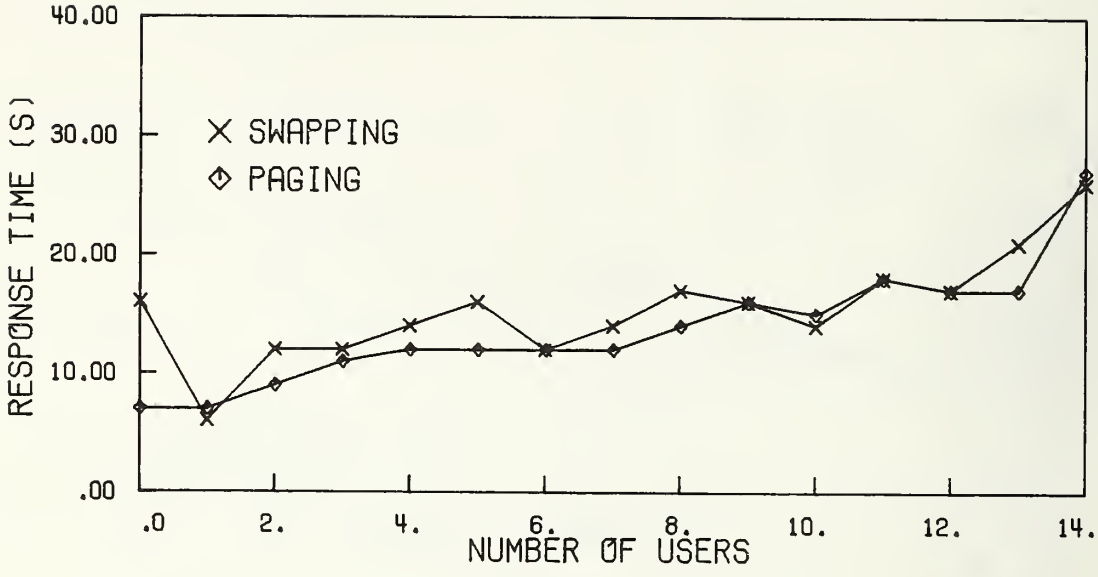


Figure 5 Response Time 75-percentile vs Number of Users for the CPU-bound job

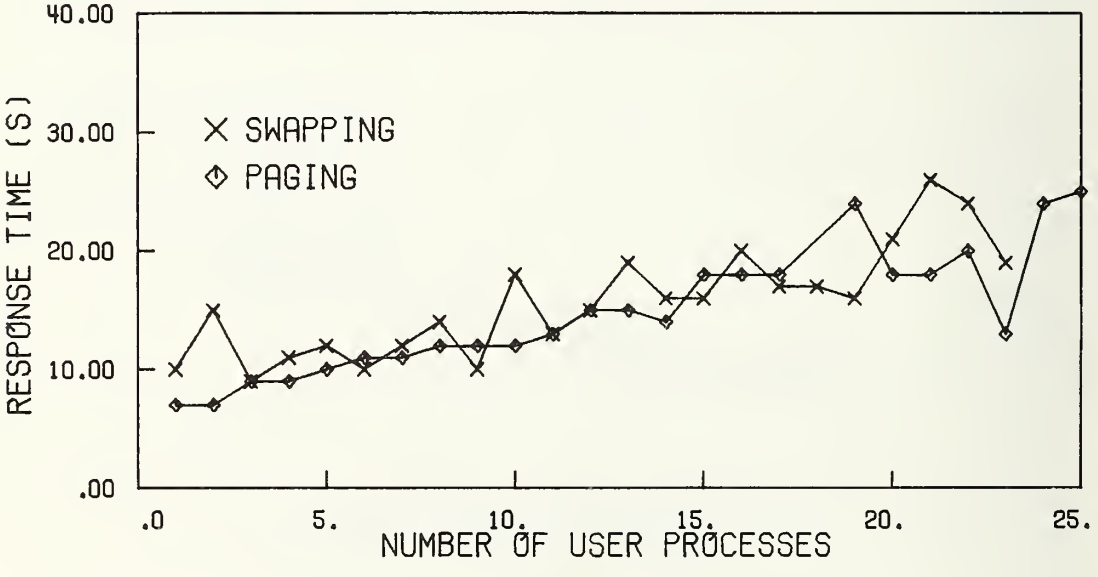


Figure 6 Response Time 75-percentile vs Number of User Processes for the CPU-bound job

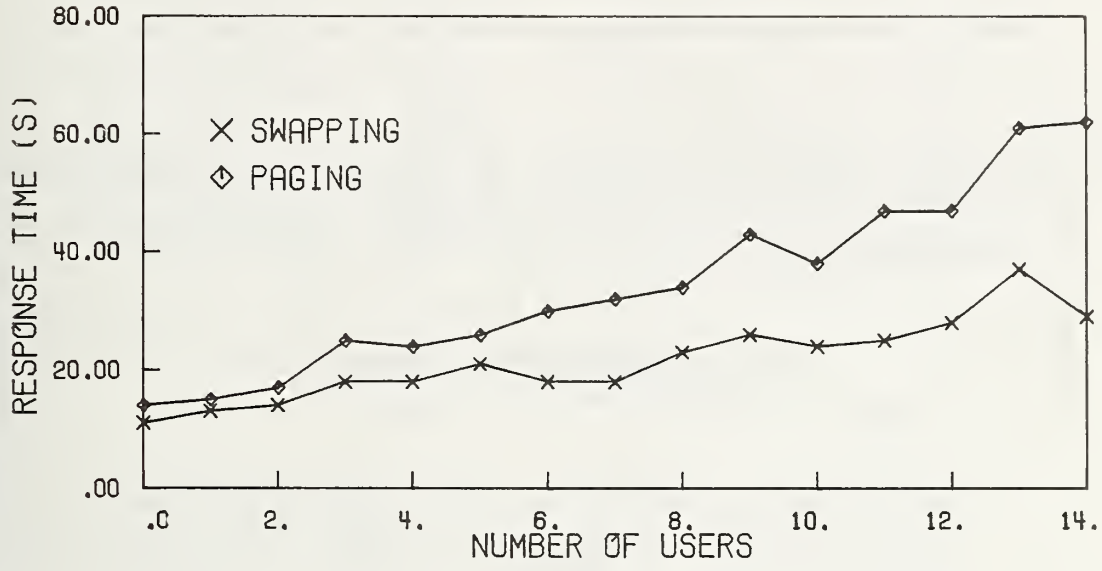


Figure 7 Response Time 75-percentile vs Number of Users for the C compilation

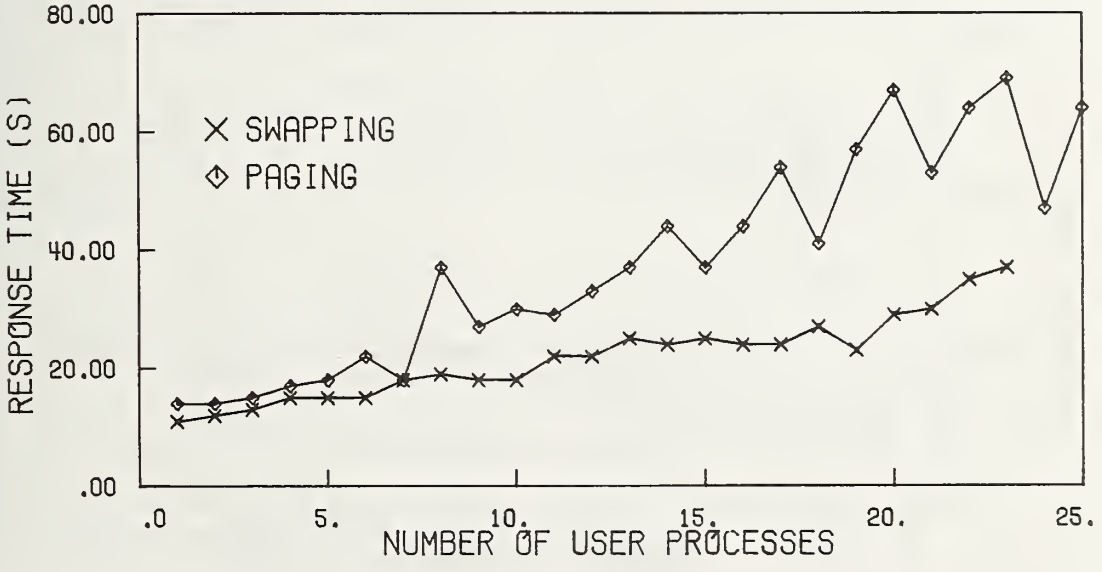


Figure 8 Response Time 75-percentile vs Number of User Processes for the C compilation

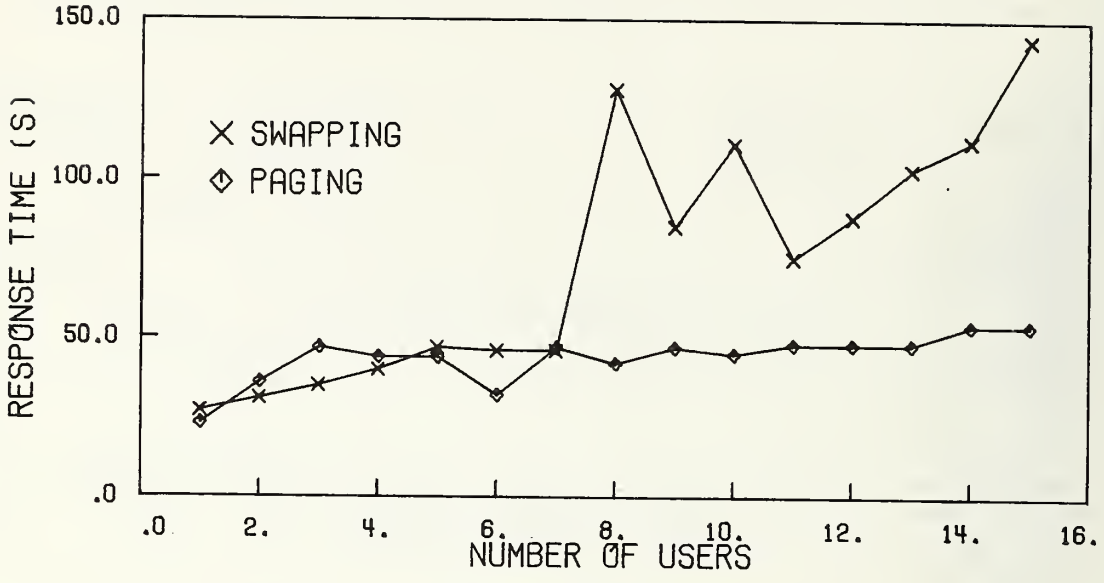


Figure 9 Response Time 75-percentile vs Number of Users for the script version of "man man"

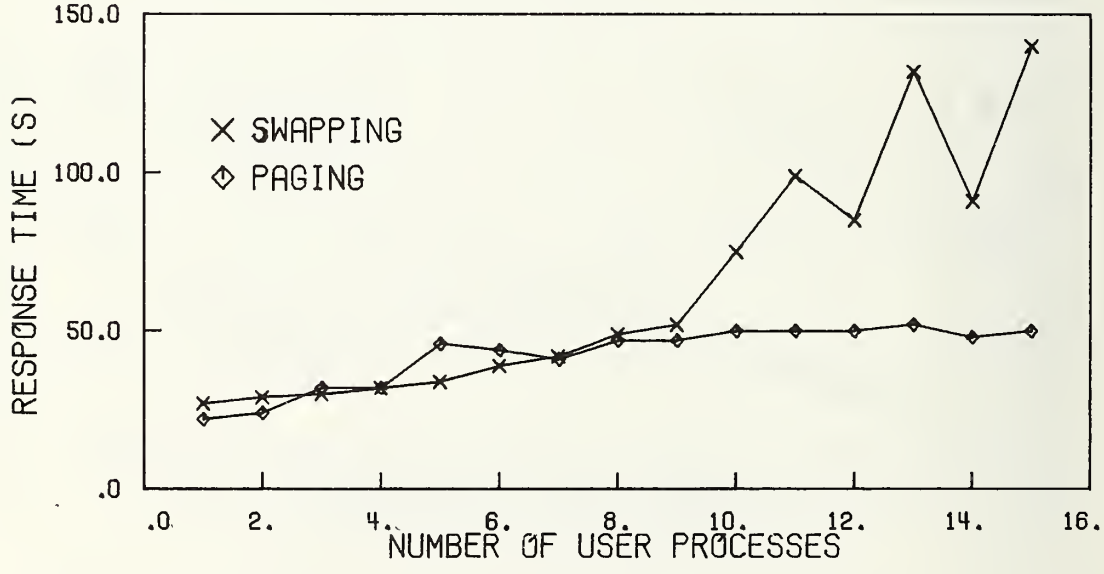


Figure 10 Response Time 75-percentile vs Number of User Processes for the script version of "man man"

PRACTICAL APPLICATION OF REMOTE TERMINAL EMULATION IN THE PHASE IV COMPETITIVE SYSTEM ACQUISITION

Deanna J. Bennett

Air Force Phase IV Program Management Office
Directorate of System Engineering
Gunter AFS, AL 36114

The Air Force's Phase IV program is the largest computer system acquisition ever attempted. It will replace the Air Force inventory of base supply UNIVAC 1050-II and base level support Burroughs B3500 computer systems. At the outset, a strategy was developed for expressing the workload requirements for the more than 200 systems to be replaced. Specific system throughput requirements were established. Along with this the decision had to be made how to validate the capabilities of the proposed systems to support the workload.

The point of this paper is to outline how the following questions were answered by the Air Force in the currently on-going Phase IV acquisition: Given a real acquisition situation, how can we test effectively? What are the practical day-to-day decisions that must be made to support the selected testing strategy?

The end decision regarding the workload test approach was to use some tools from the classic benchmarking tradition, but to enhance their effectiveness with the use of remote terminal emulation. This determination was only a starting point for a number of implementation decisions that had to be made by the Air Force to support effective use of an emulator.

The current status of the Phase IV program is that contracts have been awarded to two contractors to compete for worldwide system replacement. The contracts require transition of five major standard Air Force on-line automated data systems (ADSs) and a small sampling of standard Air Force batch systems, and demonstration of contractor-proposed systems with the transitioned Air Force standard software. At the end of the contract, based on cost and evaluation of performance, one contractor will be selected for worldwide system replacement.

Key words: Benchmark testing; Phase IV; remote terminal emulation; response time; terminal networks; workload sizing.

1. Introduction

The Phase IV program was formally established in early 1976, as the vehicle by which a large inventory of nearly obsolete computers in the Air Force inventory could be replaced. Because of its size and complexity, the Program was established on an innovative basis, unhampered by precedents in the competitive computer selection and acquisition process. The replacement strategy was developed to fit the need and

to recognize the economies of scale that could devolve from such a massive acquisition. In its overall management and procurement approach, Phase IV falls somewhere between off-the-shelf computer acquisitions typically managed by the Air Force Computer Acquisition Center (AFCAC), and the weapons system development and acquisition projects normally managed by system program offices (SPOs) throughout the Air Force.

One of the concepts typically implemented by SPOs is "fly before you buy," in which prototypes, including aircraft, are put through their paces in a real life environment before any commitment is made to large-scale manufacture. This was the most attractive aspect of the weapons system approach for Phase IV and was incorporated into our acquisition strategy. This is why we have two "prototype" system contractors. The main advantage is that it minimizes the risk of incorrectly estimating the processing capabilities of proposed computer systems before using them to replace over 200 Air Force systems around the world. A miscalculation could have severe impact on the capability of the new systems to keep up with day-to-day base mission support demands. One basic testing requirement emerged as paramount: workload processing capabilities had to be validated well in advance of commitment to worldwide replacement by either of the competing contractors.

2. The Communications Testing Problem

In Phase IV, we are using classic benchmarking techniques to create inputs to exercise the batch workload profile on the proposed system. This is necessary to verify batch processing throughput capabilities, but is not sufficient to satisfy us that the real range of workload can be supported. The base level systems being replaced through the Phase IV program have a critical on-line as well as a batch mode support roles.

There are five major on-line automated data systems which support aircraft maintenance, supply, civilian and military personnel, accounting and finance, and civil engineering functional users at each Air Force base. Depending on the location, anywhere from twelve to over one hundred fifty independent devices have to be supported by the new systems at some time in the system life. In addition to the physical capability to support the terminals, the system's responsiveness to the user is a key concern.

We rapidly concluded that batch benchmark techniques were inapplicable to our problem, and searched the field for the proper tool to apply. All the historical options for demonstrating processing capability on-line testing problems, bearing in mind the bounds of technical, economic, and practical feasibility, as well as the ultimate objective: to verify not only that the required terminal network could be supported but also that the support was timely.

2.1. Simulation Unacceptable

The verification required for on-line system sizing dictated that the actual system be tested and that the mode of test not interfere with the internal system processing. On this count alone, a simulation or model-based verification was rejected. Construction of proper models would be expensive. The technical feasibility of performing the effort was assessed as minimal, given the short time available, the range of workloads that would have to be modeled, and the likelihood that newly announced equipment lines could be proposed. Also, for all practical purposes, since the contractor would have to be performing the simulation/modeling exercise for the final test, the Air Force assurance of accuracy would come down to "trust me" as the verification that the model was performing what was required. A convincing bottom line to the brief consideration of simulation is the GSA prohibition of simulation as the sole basis for a Contract Award, a tacit acknowledgement of simulation's more artistic than empirical nature.

2.2 No Sample Terminals or Alternative Inputs

Before computers became sophisticated enough to handle more than a few terminals, the verification of a system's capability to handle a small number of terminals could be done simply by hooking the actual terminals up to the test system. As time and technology progressed, often a few terminals were used to represent a much larger whole in the benchmark process. The risk increases, of course, with the dwindling percentage of terminals actually demonstrated. After a point, the contention and queuing problems attendant on a large number of terminals are unrealistically overlooked when three or four "representative devices" are used in their place. Also, use of live terminals implies the use of live operators. While computers can generally be relied on to produce reproducible results,

human operators cannot be counted on to do the same. This violates one of the basic tenets of benchmarking in the Government: Have a controllable, reproducible experiment.

The controllability factor led to other experiments in stress testing, such as that used several years ago in the World Wide Military Command and Control System (WWMCCS) computer system acquisition. In this benchmark, the externally-generated terminal inputs were queued on tapes and batched in. While controllability and reproducibility were assured, the system impact of handling large volumes of devices and their communications overhead aspects was not assessable.

Using a combination of input messages queued on tape and a sampling of inputs from several live devices was judged to have too many of the disadvantages of either technique, and not enough advantages to reasonably pursue for Phase IV's purpose.

2.3 The Practical Default.

The option of having a full-scale benchmark test with an entire live terminal network was also rejected out of hand. The sheer management burden of monitoring and controlling activities across over 100 terminals was enough to render this practically and economically unfeasible. Dissatisfaction with the traditional choices for assessing workload processing capacity led to a single reasonable option: remote terminal emulation. The only method that could satisfy the requirement for testing throughput capacity in a terminal-oriented environment on all practical, economic and technical counts was to require a remote terminal emulator. Emulation is within the practical, technical state of the art. It is controllable and reproducible. It can demonstrate all the effects of communication contention and line strategies on the system under test. We could see no other way of testing.

2.4 Support for RTE-Based Test Requirements

Phase IV Program management was also reasonably convinced that as long as we were breaking new ground in other ways, we should take advantage of the current technology and do as complete a job of sizing verification as possible by requiring a remote terminal emulator (RTE) as a basic test tool. Before committing to an RTE requirement, Phase IV had to verify its legality in terms of Government contracting regulations.

Before the publication of the GSA RTE standard which applies across the government, specially-created, tailored variants of emulators were required when emulation was considered for application to a benchmarking problem. The variants depended on the specific desires of the individual user or acquisition office. The result was repeated development costs for competing vendors, seen as limiting the field of competition to only "wealthy" competitors. Because of this cost aspect, RTE-based pre-award competition was not permitted by GSA for Government acquisitions.

In the design of the Phase IV acquisition strategy, we circumvented this historical objection to the sunk pre-award cost of RTE-based competition. The Phase IV competing contractors will perform post-award benchmarks under paid contract while they compete for final selection as the Air Force-wide implementation contractor. Cost to vendors was not an issue, because the cost was to the Government. Because of the large number of systems involved, this investment for Phase IV is cost effective, whereas in single site acquisitions it could have been prohibitive. However, we did some groundwork before RFP release to assure our management and ourselves that our costs would be primarily test-related costs, not RTE development costs: A survey of the emulation capabilities of computer manufacturers who could reasonably be expected to submit Phase IV proposals verified that each had an existing emulator with the basic required characteristics.

The Request for Proposal (RFP) that was issued in 1978 for Phase IV was the first Air Force RFP which required remote terminal emulation. It was written before publication of the GSA standard, but it should be noted that the Phase IV RTE requirements are relatively close to the new standard.

3. Phase IV Workload Test Approach

The Phase IV problem was to orchestrate a throughput test to verify that the work of 237 separate systems with 237 variations on a common processing framework could be handled by new systems. This required development of some method for distilling the critical characteristics of the workload into simple, testable quantities, and deciding how to choose what the physical system under test should look like.

3.1 Workload Expression

System workload representation for replacement of a single system is done most commonly by constructing a single line projection of future workload and constructing a single timed benchmark test, a set of programs which reflect the work profile for the current system. (Naturally, the validity of the ensuing test is limited by the validity of the analysis and the availability of programs in appropriate languages to form the mix.) Capability to handle workload growth can then be measured by decreasing the amount of time permitted to run the program mix. Figure 1 shows this approach diagrammatically.

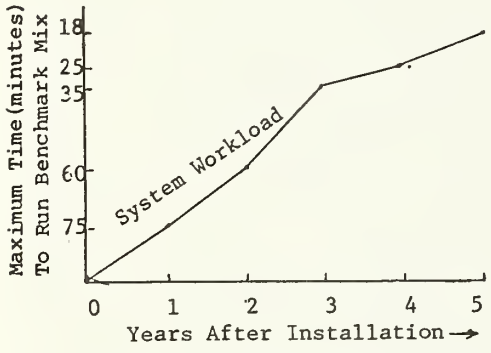


Figure 1. Typical Single System Workload Expression

To use this most common workload representation for Phase IV's wide range of types and volumes of processing could require 237 separate workload projections and tests to be developed. Besides implying an impossible test development workload, this would be spurious precision, given the basic "plus or minus" imprecisions of workload projections, however carefully they are constructed.

For Phase IV, the performance statistics from a sampling of current sites were collected and massaged into a set of workload "categories", using both the empirical data from sites and overall growth rate projections. These categories were defined to the analytical level required by sizing models. See Table 1 for an example of the workload characteristics, by category, for the U1050 workloads.

TABLE 1. X1 AVERAGE PEAK HOUR ON-LINE WORKLOAD CATEGORIES

CATEGORY	INPUT MSG	INPUT CHAR	OUTPUT MSG	OUTPUT CHAR	LOGICAL DSK I/O
1	825	72,891	5,058	450,418	42,735
2	1,817	161,713	12,017	1,069,513	94,794
3	2,556	225,027	15,583	1,387,839	132,400
4	3,218	283,035	19,590	1,744,843	166,692
5	3,728	330,729	24,193	2,151,753	205,769

Note that as these categories were developed on the basis of collections from individual Air Force bases, there is no straight line relationship across categories for various workload components. This means that test development must be tailored to each category. The single system acquisition approach of developing a single mix and decreasing the amount of time permitted for its execution to test increased throughput requirements is doubly inapplicable for Phase IV. First, it is a batch-oriented approach which does not account for variances in numbers of terminals. Second, in Phase IV, each category is really a separate workload profile, unrelated to lower or higher workload categories.

Instead of projecting workload by base, fixed categories are used and the Air Force base sites move through the categories. This simplifies the vendor sizing effort, as workload factors are already distilled into usable form for input to sizing models. Figures 2 and 3 give an example of how bases might move through the categories between two given calendar years.

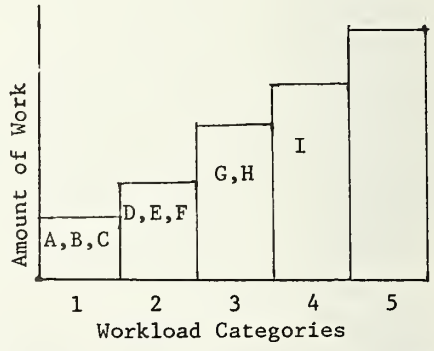


Figure 2. Year 1 Workload for Bases A, B, C, D, E, F, G, H

In Figure 2, Air Force bases A through H are mapped into 5 workload categories, de-

pending on their workloads in Year 1 of system life.

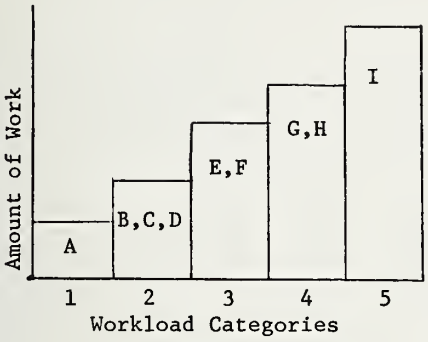


Figure 3. Year 3 workload for Bases A, B, C, D, E, F, G, H

Figure 3 shows these same bases and workload categories in Year 3 of system life, and shows how the workloads of bases have grown, causing most to migrate into larger categories.

Given this analytical framework, the testing problem is to verify that there is a system available to satisfy the workload requirements of each category, not each individual base. These category examples are simplifications. In fact there are five workload categories for the U1050 workload projects projections and 10 for the B3500 workloads.

3.2 Stress Testing Philosophy

Phase IV contractors are permitted by the Air Force to match their equipment lines against the workload categories, letting "break points" fall where they make optimal use of equipment capabilities. For example, a vendor could propose to satisfy workload categories 1 through 4 with the lowest level processor in an equipment line, categories 5 through 7 with a slightly enhanced model, and categories 8 through 10 with yet a more powerful model. Another proposal could have far different matching of the Government's workload categorizations to equipment models.

A flexible workload testing strategy was required to match the latitude allowed the proposing vendors. Stress testing was at the heart of the strategy: the workload processing capabilities tested had to be measured by testing the largest workload against which a configuration was proposed. This required us to be prepared to test any of the fifteen workload categories we had established to anticipate all potential "break points" in contractor proposals. For the Air Force to develop a test for each category and mode, as we in fact did, required construction of 32 separate tests -- 15 on-line, 15 batch and 2 combined on-line and batch tests.

One major feature of the Phase IV acquisition that made such a test development effort feasible is that the contractor will be supplying all the software for the test. All workload categories and their characteristics (messages in, characters in, cards punched, and so forth) are expressed in terms of automated data systems (ADSs) which the contractor is responsible under contract to transition to his proposed system. The Air Force test development effort did not require selecting a specific mix of high-order language programs that the competing contractors could easily run on their systems, as in historical benchmark construction. Rather, a carefully measured collection of transactions was collected to exercise the collection of ADSs to fit the workload profile. This applies equally to the batch and on-line tests (for Phase IV separate on-line and batch workload tests are required except in a few cases); however, this paper addresses solely the on-line test construction problems. The amount of work done by test developers in selecting the individual program transactions from this large pool was monumental, but was unhampered by the "high order language" constraint.

4. Specific RTE Implementation Decisions

The Phase IV on-line workload tests are executed by the Phase IV contractors on their own RTEs using Air Force software they have transitioned to work on their systems. Much of the test burden is put on the contractors. Commitment to the use of an RTE for validating terminal-related processing capabilities did not remove all the Air Force test burden, nor was it an end decision. It was only the beginning of a series of other implementation decisions.

4.1 How Many Terminals To Test?

Given the workload definition scheme and the determination that an RTE would be required, the next decision to round out the definition of the test environment was how many terminals would be associated with the on-line test for each workload category. The RFP presents two terminal profiles for

each system at each base: the number and types of terminals required at initial installation, and the number and types projected at an augmentation point 54 months after initial installation, a point close to the end of the initial contract life. We took the obvious option and decided to test the larger number of terminals, that at augmentation.

Many Air Force base systems are associated with each workload category at the augmentation point. We had to test the terminal configuration for an actual Air Force base as defined in the RFP because this is the basis for vendor proposals, and, most importantly, for their communications strategies. Our stylized workload categories associate the same number of on-line transactions with all bases within the category, independent of the number of terminals actually projected for the base. We developed tests for the terminal profile of the base in the category with the largest number of terminals. This tests the fullest communications-related overhead, e.g., polling, multiplexing, and/or concentration (depending on the contractor-proposed methodology), for the largest number of terminal devices. If a configuration can support the largest number of terminals projected for it, then it certainly can support fewer terminals.

The decision involved a trade-off. There is a Phase IV terminal response time requirement as well as a workload volume requirement. Spreading inputs and their associated outputs across a large number of terminals could have the effect of making the average terminal response time easier to meet than if the same number of transactions had to be queued up for fewer terminals. The risk of making the response time easier to meet was far lower than the risk of specifying a test system with so few terminals that we wouldn't be prepared to validate a unique communication strategy a vendor might propose for larger numbers of terminals.

4.2 Who Provides the RTE Inputs?

The contractor-transitioned Air Force software was the basis for the test. But who would create the emulated terminal inputs?

We could have defined a "mix" of transaction types for each transitioned data system, (e.g., 40% updates, 50% inquiries, and 10% record deletions), and had the contractor provide the transactions. The problem with this is that the system test activity must reflect the activity defined in the Phase IV workload category definitions for the test to produce valid results. Gross definitions of transaction types could in fact result in the contractor being forced to generate more workload than required by workload sizing merely to fit the transaction percentage profile--or, in imposing lesser workloads than required because the contractor could select low activity variants of transaction types.

Another "easy" alternative from the Government test development standpoint is to put the entire burden on the contractor and let him run any mix of transactions which he says will fit our "bottom line" workload category requirements. The additional test burden is on the contractor -- but there is virtually no way in which the Air Force can be assured that what we see is what we think we are seeing!

These two alternatives have been used in other RTE-based system tests. They generally stem from a desire to see the system driven, but a lack of software on which the user can base the demonstration, either because of the cost and time to convert extant complex interactive software, or because the teleprocessing requirement is new and there is no software. Phase IV is not constrained by the lack of demonstration software. Since the contractor-converted Air Force automated data systems have to accept the same inputs as on current equipment, we decided to specify every transaction for every emulated terminal. We could and did develop individual terminal scripts whose "bottom line" total workload fit the profiles for each workload category. Note that both the terminal activities and the initial workload categorization were based on our current equipment, so the test and specification are totally compatible.

4.3 What Transactions to Select?

Methodology for test transaction selection was to individually test a small sampling of transactions from all software systems that were part of the on-line workload test to determine their profiles when acting against pre-defined test data bases. This profile included number of disk I/Os generated, number of characters input and output, and other characteristics directly related to the workload category figures. Then an automated "mix and match" program selected the proper numbers and kinds of transactions and created the speci-

fic scripts by terminal by software system (ADS). Each script was validated to have the proper number of transactions and in total they were validated to provide the correct amount of work against the ADS.

Besides completion of all specified processing in one hour, there is a response time requirement to be met in the workload test. The Phase IV systems must provide a maximum 10-second average response time for 80% of all on-line transactions, with a maximum 30-second response time for any one on-line transaction. The sampling of transaction types used in the Air Force scripts was carefully selected to support a fair test of on-line support capability. Real, on-line, interactive type transactions were selected. Batch-oriented transactions were not included on the rationale that response time's importance is related to queries for small quantities of information, not large reports. A batch job which requires significant file manipulation is not an interactive type process just because it can be activated from an interactive terminal.

Because of the average response time requirement, we had to further restrict our test transaction types and test data. For Phase IV's workload test, response time is defined as the time between the last operator action of the input and the display of the first human-readable character of the output. What if there is no response? What if the input generates output to several terminals? In our case, if that occurs, it invalidates our average response time calculations. You can't calculate the time between input and output if there is no output, or if there is no input. As these types of transactions are supported by current Air Force software that will be carried over to the new system, the transactions that exhibited unacceptable forms of output for test purposes and response time calculation were eliminated from consideration in building our terminal scripts.

4.4 Who Controls Timing of RTE Inputs?

The last critical strategic decision to be made was how to handle time spacing of the inputs. Our terminal scripts spread inputs for each terminal across the 60-minute test period, ensuring that the last input was entered sufficiently in advance of the end of the 60.00 minutes that its response could be generated within the timed test period. Further, the scripts for individual terminals interacting with the same software system were given staggered start times within the first few minutes of the timed test period to avoid a "lock step" series of contention-generated delays.

Though our scripts were generated so they could be run "as is", we had to decide if that should be required of the contractor. Execution completely in accordance with the script time spacing of transactions might make it impossible for the contractor to meet response time requirements. This is because all the scripts for all the software systems in the mix could not be tested in advance, at the same time. (If we could do that, we would be able to handle our projected workload on the systems we have today, and we wouldn't need to replace them!) There is a potential that some combinations of transactions across ADSs and terminals might unintentionally occur in such a way that a snowball attenuation of response times could result. We decided that the contractor shouldn't have to pay for the quirks in our script development effort, and that some adjustments to the timing of inputs should be permitted.

The degree of freedom permitted the contractor was important to define. If we gave them complete control, there would be nothing to prevent a virtual non-mix to be run, with the terminal scripts run sequentially on an ADS by ADS basis, e.g., running all supply transactions, then all finance transactions, etc. Our ultimate decision was that the contractor could run all scripts as provided, or could modify them under tight constraints.

If the time delay between inputs from any script is to be adjusted by the contractor, the script must be changed so that the first transaction is initiated in the first five minutes of the test period, the last transaction is entered in the last five minutes, and all transactions in between are distributed evenly across the test period, i.e., they have identical delay times between them.

5. Resulting Test Characteristics

What we ended up with at the end of this on-line test development effort was the following:

(1) A useful test tool, a remote terminal emulator, defined as the basis for the test.

(2) A workload test strategy that

rested completely on the pre-defined workload categorizations which could verify the telecommunications support capability by the system under test in the most stressful (largest number of terminals) situation.

(3) Scripted terminal inputs and outputs designed by the Air Force to specifically exercise the Air Force-defined workload.

(4) A strategy for permitting the contractors to eliminate timing-related quirks in the Air Force data without degrading the purposes of the mix test.

What we didn't fully realize was that we also had a flexible test basis with which we could test the workload processing support capabilities for any of the Air Force sites in the world -- with some test degradation. This "object lesson" came toward the end of the development cycle for the thirty-two separate Phase IV workload test packages.

6. Accounting for Implementation Redirection

Original Phase IV plans called for one-for-one replacement of current U1050 and B3500 systems, even though they usually are located on the same base. After initial Phase IV proposals were received and after a large amount of work had been invested in developing workload tests, Air Force and congressional reexamination of Phase IV's implementation plans resulted in a redirection of this concept. The one-for-one philosophy fell by the wayside for a large number of bases which would now either get a single computer system to replace their two on-base computers or be remote sites associated with regional centers. The same workload categories applied, in additive fashion, to the single systems: the U1050 and B3500 projected workloads at a given Air Force base would have to be handled by the same system.

From the batch testing standpoint, no real problems were posed. The U1050-based test and the B3500-based tests could be run concurrently to test capability to process the combined workloads. From the terminal network and testing stand-point, at this point all bets were off. The workloads of the two systems at each base are independent. The mix of new combined workloads and remaining single workload systems meant that workloads from separate categories were combined on a base-by-base basis. The "maximum number of terminals" approach didn't fit the real life bases anymore. The base with the largest number of terminals for its B3500 workload category could have far less than the maximum number of terminals for its U1050 workload category. We couldn't run the RTE tests "as is" because we would be requiring the contractors to test networks that were not proposed and not required by the contract.

The problem test designers had to deal with was simply put: Now that the tests have been designed and developed to spread all the on-line work across the maximum number of terminals for any base in a given workload category, how can we test fewer.

One option was to redevelop the tests. This was impossible. While we were updating our strategy, the offerors were updating their proposals. Having to deal with the proposal flexibility factor would put us back in the mode of developing over a hundred tests to cover all possible combinations, just to be sure we had the right mix available for the contractors to run.

The other option was to maintain the amount of work, but decrease the number of emulated test terminals to fit the actual test Air Force base. Decreasing the number of terminals to be emulated would apparently also lose the work input by these terminals. Rather than "falling back" on the amount of workload required to be processed, the work was input directly at the mainframe instead of via the deleted terminals. This could be done because of the way the Air Force software works. Live terminal transactions can be batched into a disk file in the test set up process, and pulled off the disk file to have the same effect on the data base as the live transactions for the B3500 software during the test. For the U1050 software, direct card reader input of the transactions identical to the live transactions has the same effect. Any terminal in the test data package in excess of the actual test configuration was deleted from the test, but its transactions were redirected for the alternative input. From the network standpoint the test is somewhat easier. From the workload standpoint, the system is performing the same test. This approach required no redevelopment of test materials, continued to support full workload testing, and verified to the developers that we had a powerful, well-constructed test methodology.

7. Conclusions

This paper has described how Phase IV decided that a remote terminal emulator would be used for its on-line workload tests and the major implementation decisions we had to make in supporting the use of the RTE. In addition to those mentioned above, we had to decide what to call our terminal input collections -- scripts as the Air Force does, or scenarios as the rest of the world does. We had to decide how fast a terminal operator types so we could include the correct operator time in our script time delays and in our constraints to the contractors. We had to weight the number of transactions that could be input and output at a device by the kind of device it was -- a terminal with a screen display displays output faster than a printer output terminal, and so can handle more transactions in an hour than the printer terminal. We had to decide how to present the scripts to the contractors, and decided that print image format on tape would be acceptable. And there were many other decisions like this, every day of the test development effort.

We planned RTE use into our on-line tests the hard way. There was no GSA standard, so we had to create our own RTE specification, and justify it as a requirement. Just about everywhere there was a decision to be made, we made the decision that would give the Air Force the most visibility and the most control over the testing process. Usually this meant more work for us. But we have the users of 237 Air Force systems to answer to if things don't work out correctly.

In the end, we decided that we have a very good on-line workload test. It hasn't been run yet by the contractors, and that will be the final proof. We also have come to realize, over the course of thousands of hours of computer time and several years of development and refinement of RTE scripts, that use of an RTE is not to be undertaken lightly. The end requirement must completely justify not only the burden put on the contractor performing the test, but also the massive amount of time and money it takes to adequately prepare for its use.

CPEUG81

Cost/Performance Evaluation – Methods and Experiences

AN EVALUATION TECHNIQUE FOR EARLY SELECTION OF COMPUTER HARDWARE

Bart D. Hodgins
and
Lyle A. Cox, Jr.

Naval Postgraduate School
Monterey, CA 93940

There is a need for a decision making tool for use early in the computer selection process. Such early selection tools are critical to the decision maker due to the environment in which the manager of large scale procurements is forced to operate.

The instruction mix technique was well known in the early years of computing. It was simple, easy to use and easy to understand. However, as the complexity of machines and support software grew, the technique was replaced by more sophisticated performance prediction tools. With certain modifications, this technique can continue to provide important information. By using a number of instruction mixes in a computer assisted environment, it is possible to find performance trends of hardware in the very early design stages. These trends can be used effectively to make selection decisions.

The instruction mix sensitivity technique as demonstrated here has the potential to aid the decision maker in evaluating the performance of a system prior to the actual existence or availability of that hardware without resorting to costly and time consuming techniques such as simulation or modeling.

1. Introduction

There is a need for a decision making/selection tool for use in the computer selection process (particularly for those involved in the very large procurements typical of our government). The instruction mix sensitivity technique as demonstrated here has the potential to aid the decision maker in evaluating the performance of a computer prior to the actual existence or availability of that hardware without resorting to costly and time consuming techniques such as simulation or modeling.

1.1 Background

Large Automated Data Processing procurements evolve through a cycle that often last five to seven years. The selection of computer hardware is forced to occur early in the procurement cycle. This long period of time from selection to operational installation often necessitates procurement decisions to be made before prototype hardware is available. Hardware selection must be made quickly and accurately. Errors cost time and money. Any delay caused by selection can have a ripple effect building through the entire process. Poor selection of the hardware to be used as the basis for a system can result in cost overruns in other areas to compensate for the lack of acceptable hardware performance. At present there is no general method for computer hardware evaluation and selection completely suitable for use early in the procurement cycle. Poor procurements are often made because the decision maker is forced to make a selection without the benefit of having candidate hardware available. Similarly, selections of equipments are often based on imprecise and quantitatively vague ideas of the actual operational utilization that the system will face years in the future.

There are several methods currently being utilized for the evaluation of a computer's performance. They include:

(1) benchmark programs which are existing programs coded in a specific language, then executed and timed on a target machine [1], (2) kernel functions which are typical functions partially or completely coded and timed [1], (3) simulations which are a combination of a model of the system, model of the workload, and a measurement of the resulting data [2], and (4) analytic models which are mathematical representations of the target machine [1]. These methods are all in use (principally in industry) to evaluate proposed computer systems for procurement. For smaller procurements these methods are effective because their acquisition cycle is shorter. In some cases it is even possible to wait until both hardware and software are available, and to use the standard evaluation techniques in making a specific computer system selection.

In large acquisitions, selection must occur early, and unique problems arise that cannot be easily solved by the various evaluation techniques. Evaluation by the benchmark program method is impossible because the different machines are not always available. Even if a prototype hardware of a future system were available for evaluation, the benchmark programs and the kernel function methods often prove inadequate because the support and applications required to validate the technique usually do not exist at that point.

When a selection must be made quickly, there is usually neither the time, money, nor the sufficiently detailed design information necessary to model/simulate the proposed computer systems. How can the selection be scientifically made? It is because of this problem that the instruction mix sensitivity technique (IMSET) has been developed.

1.2 The Instruction Mix Sensitivity Technique Overview

The instruction mix sensitivity technique is based upon the older instruction mix method for predicting computer hardware performance. In the instruction mix method a number was computed which represented the average thruput of a particular hardware. This number was based upon the relative usage of a given instruction in a particular application, and its execution time on the evaluated hardware. Where the older method was based on a single mix representing a specific application, the sensitivity technique evaluates trends in the differences between a computer's predicted thruput over a collection of mixes representing various applications. The advantage of this technique is that neither the hardware of software need be completed -- only the organization and technology need be determined. The eventual utilization of the system need not be precisely defined. This technique provides immediate evaluation results with a minimum expenditure of time and money.

Using the IMSET requires only that the vendor furnish the performance specifications regarding instruction execution times. These performance specifications are often available years in advance of a prototype model. With these times, and the analysis technique presented here, the evaluator can compare the predicted performance of any hardware against the anticipated application. The particular machines to be considered in the selection need not be prototyped.

The use of the IMSET as a tool for evaluation provides the decision maker with a profile representing the candidate computer's average execution time for the various applications presented in the set of instruction mixes. From the data the decision maker can select the hardware with the best profile for the intended application. For example, if the evaluator is looking for a machine to perform accounting functions then the selection would be based upon how sensitive each candidate is to the mixes which represent accounting and related functions. The less sensitive the machine in terms of execution time the more appropriate it would be for selection, since this indicates that it can execute effectively a broad spectrum of related functions.

2. Instruction Mix Sensitivity Technique

2.1 History

The instruction mix as a technique for evaluating the performance of a computer's hardware came into being in the late 1950's and early 1960's. It evolved as a result of the limitations of an earlier technique for measuring a computer's performance called the instruction execution timing method.

The instruction execution timing method considered only a single class of instructions. The instruction mix technique incorporated along with the arithmetic class, the logical class (i.e. COM-

PARE, SHIFT, MOVE, etc.), and in some instances I/O and other miscellaneous instructions. Associated with each instruction in the mix was the percentage of use of that instruction, called a weighting factor unique to that particular mix. This weighting factor represented the approximate probability of occurrence of that instruction in the programs to be used on the machine. For instance, in a scientific instruction mix one would find that the percentage of floating point multiplications would be higher than the percentage for that same instruction in a data processing instruction mix. The probabilities in an instruction mix were determined by either statically or dynamically tracing the programs representative of specific applications.

The instruction mix technique is easy to apply. By multiplying the execution time of each instruction by the weighting factor and summing, one obtains the time required to execute an average instruction for that particular mix on that particular computer. This average time can be inversely related to a thruput rate in instructions-per-second (IPS). These totals can then be compared with similar rates obtained from other machines, to give an idea of relative CPU thruput.

Mixes for many applications have been developed. The most popular of all mixes was the Gibson Mix [3] developed by Jack C. Gibson in 1959 on data obtained on the IBM 7090 computer. The Gibson Mix was considered a general-technical mix. There were other similar mixes [4, 5, 6, 7, 8, 9, 10] for data processing, navigation, scientific, and a myriad of other applications.

As computer hardware and software technology advanced, especially as systems moved into a multiprogramming environment, it soon became apparent that using a single instruction mix to evaluate absolute performance was no longer adequate. Among the shortcomings of the technique was its failure to account for differences in addressing modes, word sizes, and operand lengths. The effect of system software upon the mix weights was difficult to assess. Perhaps the biggest disadvantage was the problem of how to validate the instruction mix to insure that a particular mix accurately reflected the intended application. How can the instruction probabilities be determined if the programs representing the eventual workload have not yet been written?

2.2 IMSET

The variation of the instruction mix technique described here is called the instruction mix sensitivity technique (IMSET) [11]. The IMSET uses a set of ten instruction mixes chosen from over 20 candidates [11] to represent a wide range of computer applications, spanning from real-time thru scientific to business computations. Utilization of the IMSET provides the evaluator with a profile representing a hardware's execution times across all mixes in the set (and hence across a broad spectrum of applications). The comparative profiles of execution times provide the decision maker with a quantitative prediction of how sensitive each computer is to the various mixes and hence how the system would perform over a wide range of applications. This is in contrast to the instruction mix technique which only provided the evaluator with a thruput evaluation on one mix -- one application.

The IMSET uses eighteen functional instructions which constitute the basis for evaluation. These include seventeen specific instructions and one "I/O miscellaneous" category. The eighteen functional instructions and ten mixes which constitute the IMSET are shown in Table I.

To use IMSET it is necessary to determine the execution times of the 18 instructions for each central processor to be evaluated.

A detailed discussion of the development of this technique and the details of calculating hardware timing figures are discussed further in Hogin's report [11]. While particular care must be paid to determining the timing figures, the procedure is not difficult.

Of particular interest is the handling of concurrency in modern hardware. In the standard application of the instruction mix technique many special features such as concurrency of a central processor's hardware were ignored. For example, some processors begin execution of a second instruction before the current instruction has finished its execution. This allows effective execution times to be cut significantly. The IMSET presented here takes into account the overlap capabilities of the machines being evaluated by applying a "Knuth Factor". This idea was provided by [12], and is described in [11]. The Knuth Factor compensates for

TABLE I

IMSET FUNCTIONAL INSTRUCTIONS AND MIXES

	I. ARITHMETIC								II. LOGICAL			III. CONTROL						IV. I/O MISC.
	FIXED POINT (SP)			FIXED POINT (DP)		FLOATING POINT (SP)							BRANCH					
IMSET MIXES	ADD/SUB	MULTIPLY	DIVIDE	ADD/SUB	MULTIPLY	ADD/SUB	MULTIPLY	DIVIDE	COMPARE	SHIFT	AND/OR	LOAD/STORE	CONDITIONAL	UNCONDITIONAL	INC & STORE INDEX	MOVE	INDEX	I/O & MISC.
PROCESS CONTROL	.064	.013	.001	0.0*	0.0*	0.0*	0.0*	0.0*	0.0	.022	0.0	.400	.480	0.0*	0.0	0.0	0.0*	.020
MESSAGE PROCESSING	.050	.005	.005	0.0*	0.0*	0.0*	0.0*	0.0*	.010	.030	.150	.470	.140	0.0*	.030	.058	0.0*	.052
REAL TIME	.126	.108	.020	.014	.012	0.0*	0.0*	0.0*	.020	.070	.010	.500	.100	0.0*	0.0*	0.0*	0.0*	.020
COMMUNICATION CNTL	.080	.002	.002	0.0*	0.0*	.005	.001	.001	.075	.075	.070	.475	0.0*	.050	.030	0.0*	.035	.090
DATA COMPRESSION	.190	.060	.060	0.0*	0.0*	0.0*	0.0*	0.0*	.120	0.0	0.0	.270	.090	0.0*	.060	.060	0.0*	.120
NAVIGATION	.230	.250	0.0	0.0*	0.0*	0.0*	0.0*	0.0*	.020	0.0	0.0	.300	.020	0.0*	.040	0.0	0.0*	.140
TLM THRUPUT	.090	.040	0.0	0.0*	0.0*	0.0*	0.0*	0.0*	.290	0.0	0.0	.220	.150	0.0*	.040	0.0	0.0*	.190
TECHNICAL GENERAL	.025	.025	.025	0.0*	0.0*	.011	.011	.011	.108	.045	0.0*	.361	.275	0.0*	0.0*	0.0*	0.0*	.103
SCIENTIFIC	0.0*	0.0*	0.0*	0.0*	0.0*	.095	.056	.020	0.0*	0.0*	0.0*	.285	.132	0.0*	0.0*	0.0*	.225	.187
COMPOSITE GENERAL	.022	.022	.022	0.0*	0.0*	.003	.003	.003	.120	.029	0.0*	.360	.358	0.0*	0.0*	0.0*	0.0*	.057

* Weight not assigned by mix for this functional instruction.

the machines which have overlap or parallel processing abilities by scaling down their execution times by an amount comparable with the typical use of this feature by most programs and compilers.

2.3 Some User Notes

The I/O instructions omitted from many mixes caused problems with early evaluations. I/O instructions are a mixture of peripheral capability and a central processor capability. The mixes presented here include the I/O instructions in the miscellaneous category rather than as a specific instruction. In this way the central processor's ability to handle I/O is treated as an average over all of the I/O instructions without having to specify an exact instruction or particular device.

It should be pointed out that the instruction mix was a tool to be used principally for the comparative evaluation of the central processor hardware. The way the central processor is configured with other system components such as storage devices and other I/O and peripheral devices must be considered separately. This characteristic of the instruction mix technique carries over to IMSET.

The user must be aware that the software associated with the system includes the operating system, language processors, in addition to applications programs. All these have an impact upon overall performance. The mixes in IMSET reflect all types of computation, making it possible to consider performance in both system software and application software areas.

3. A Demonstration of IMSET

In the final stage of the IMSET development process six micro-computers were selected for competitive evaluation to demonstrate the strength of the IMSET when used to evaluate machines closely related in characteristics. This was intended to reflect an actual evaluation for selection situation. The micros selected are all 8-bit or 16-bit machines ranging from some earlier models to some much more recent ones. Those selected are the ZILOG 8000, INTEL 8086, MOTOROLA 68000, DIGITAL EQUIP. CORP. LSI 11/23, and the TEXAS INST. 9900.

Determination of the individual instruction times for each of the micro-computers is presented in [11].

The demonstration to obtain the profiles of all hardwares chosen versus the set of ten mix applications of the IMSET was conducted on the computerized evaluation system. Three sample profiles are shown in Figures 1 through Figure 3. Figure 4 is the composite of all the micros. Table II provides a key for the instruction mixes listed by letter for each of the computer profiles.

When analyzing the profiles it is important to remember that the purpose of the IMSET is to compare a machine's execution time sensitivity between applications, not only its estimated effective execution speed for any one application. The sensitivity between applications is determined by comparing the times of execution -- the performance trends. Consider the following example:

Table II

KEY TO THE INSTRUCTION MIXES PRESENTED IN FIGURE 1 THROUGH FIGURE 4

LETTER	INSTRUCTION MIX
a	Process Control
b	Message Processing
c	Real Time
d	Communication Control
e	Data Compression
f	Navigation
g	TLM Thruput
h	General Technical
i	Scientific
j	General Composite

Suppose we wish to select a microcomputer whose principal purpose is expected to be the processing of both navigation and telemetry applications.

We first screen all candidate machines to verify that they 1) meet our

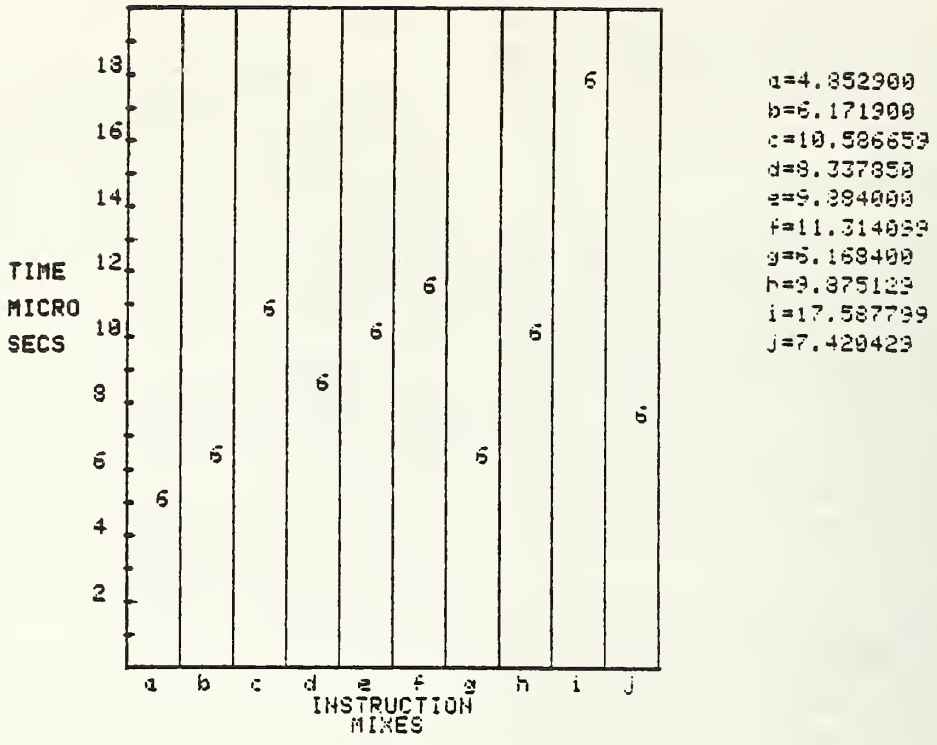


FIGURE 1. DIGITAL EQUIPMENT CORP LSI 11/23

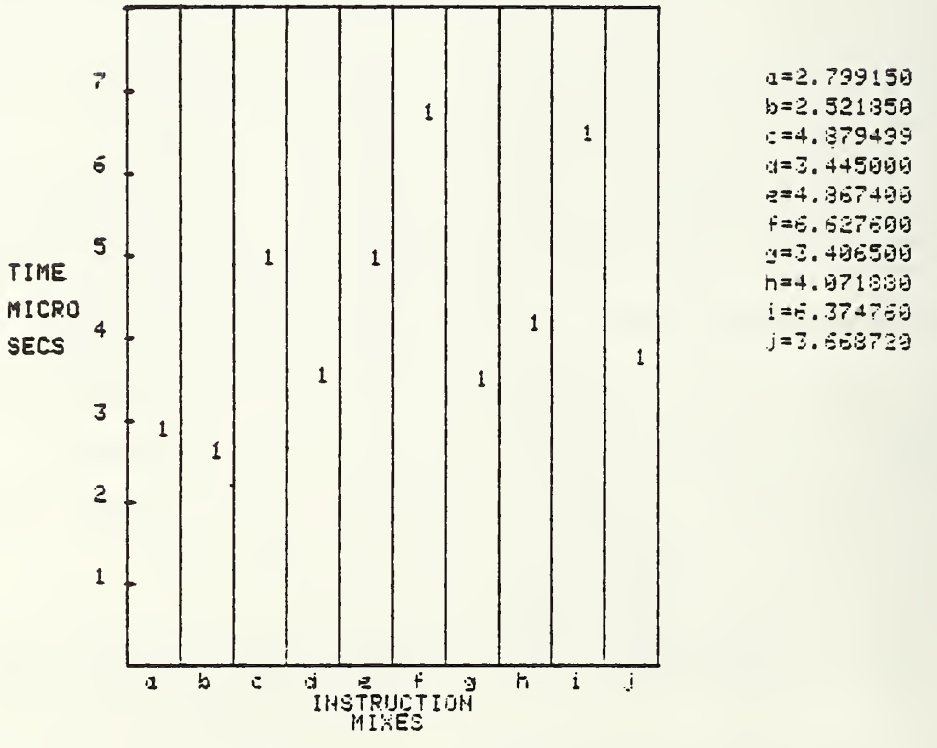


FIGURE 2. ZILOG 8000

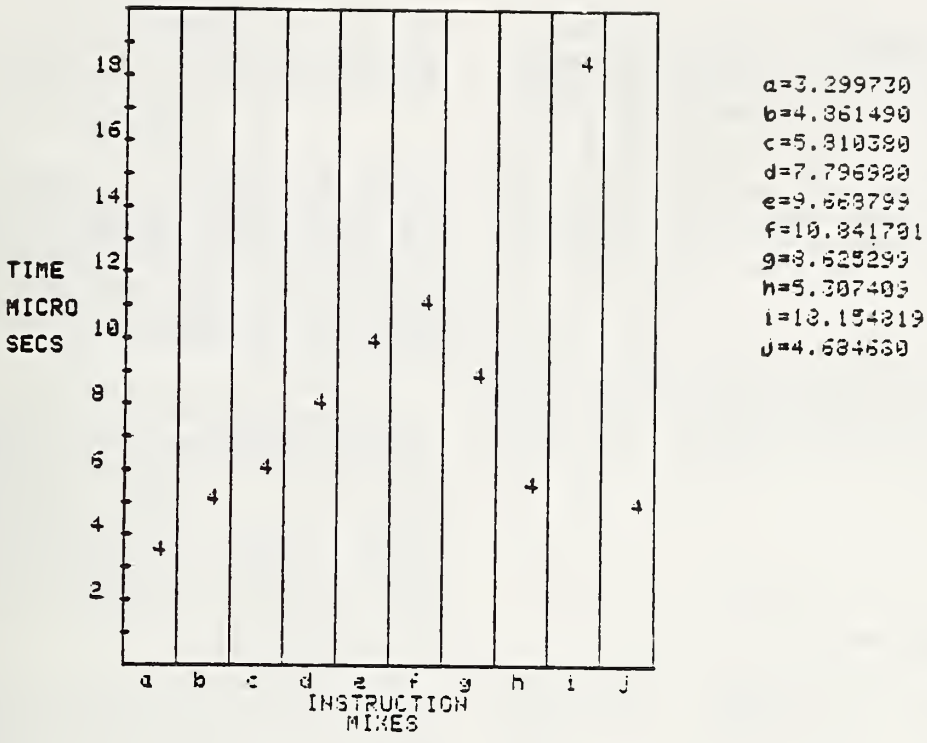


FIGURE 3. MOTOROLA 68000

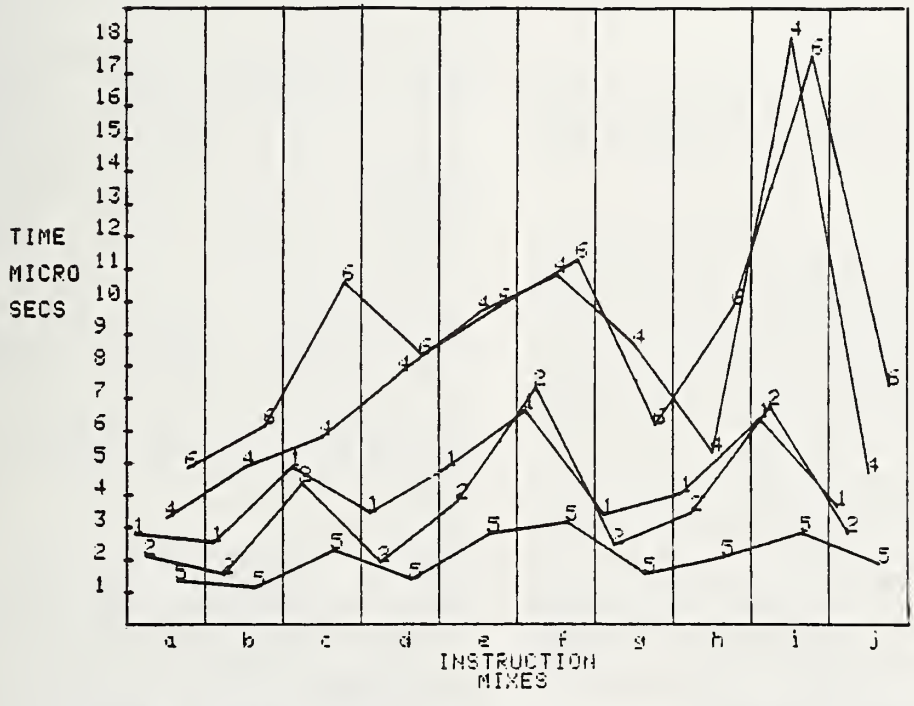


Figure 4. Composite sensitivity profiles of microcomputer hardwares:
(1) ZILOG 8000, (2) INTEL 8086, (4) MOTOROLA 68000,
(5) TEXAS INSTRUMENTS 9900, (6) LSI 11/23.

minimum absolute performance requirements and 2) conform to other system requirements such as availability of support software, peripheral compatibility, etc. For those machines which meet these criteria, we apply the IMSET procedure. The micro-computer selected should present the smallest change between the two applications (since the eventual percentage of workload is not known for the two components).

For our example, we will assume that the IMSET technique will be used to decide between the Digital Equipment Corp. LSI 11/23 and the Motorola 68000 for our navigation and telemetry problem. The IMSET results are:

Table III

IMSET Demonstration Results

PROCESSOR	Avg. time/inst. (microseconds) NAV Mix	TLM MIX	Sensitivity (per cent)
Motorola 68000	10.8	8.6	26
DEC PDP 11/23	11.3	6.2	84

This analysis suggests that the MOTOROLA 68000 might be the more preferable micro-computer due to its lower sensitivity (more uniform performance) to the two applications.

If we were certain that the principal function of the system was telemetry, that the navigation functions to be performed are minimal, and that this balance between the functions would never change significantly during the lifetime of the system, the DEC 11/23 would be the better choice due to its greater thruput for telemetry applications. In the early stages of system design we are seldom sure of all these things. Averaged over both applications, the performance of the two candidates is roughly equal. However, as long as we cannot be certain of what the workload proportions will be between navigation and telemetry, it would be prudent to select the machine whose performance is less sensitive to fluctuations in the job mix, in this case the Motorola 68000.

This is a very simple example. The IMSET is capable of being used in much more challenging evaluation situations, with more complex workloads and complicated computer systems. It does, however, demonstrate its utility and ease of use.

5. Conclusions

We have introduced a method, the IMSET, with which the decision maker can quickly and efficiently select a computer hardware from a number of candidates. One simple example of the application of this method was presented and profiles of actual hardwares were shown.

Validation of the mixes vs. applications when using the IMSET for evaluation is not the problem it was for the instruction mix method. When evaluating by the IMSET method, particular sensitivities within a broad area of intended use are being measured, whereas with the instruction mix method, execution time of a specific application was being estimated. Thus, validation is not an issue when utilizing the IMSET.

The advantages to using the IMSET over other currently used techniques are tremendous. As mentioned above when early selection is required, the decision makers are often not able to utilize many of the more sophisticated, and proven methods. With the IMSET the decision maker needs only the manufacturer's projected instruction set execution times. With these times it is possible to obtain the evaluation data within a matter of hours and at minimal cost. This technique provides a savings in time, savings in money, and greater confidence in the selection. Perhaps its most attractive advantage is its ease of use.

The data provided by IMSET can be integrated easily into more comprehensive selection techniques. While a discussion of this aspect is beyond the scope of this paper, it is easy to see how IMSET could be used in specifying or selecting systems when used in the context of the Cost-Value Selection Technique [14], or similar methodologies.

The strength of the IMSET as an evaluation tool lies in the fact that it is able to be applied in the absence of available hardware and specific knowledge of intended application. It is very important that a tool such as the IMSET be an integral part of any decision making process affecting the procurement of computer systems in the future.

REFERENCES

1. Lucas, Jr. H. C., "Performance Evaluation and Monitoring," Computing Surveys, V. 3, No. 3, P. 79-90.

2. Svobodova, L., Computer Performance Measurement and Evaluation Methods: Analysis and Applications, p. 52-56, American Elsevier, 1976.

3. IBM Technical Report TR00.2043, The Gibson Mix, by J. C. Gibson, p. 2, 18 June 1970.

4. Corsiglia, J., "Matching Computers to the Job - First Step Towards Selection," Data Processing Magazine, p. 23-27, December 1970.

5. Arbuckle, R. A., "Computer Analysis and Thruput Analysis," Computer and Automation, V. 15, No. 1, p. 12-19, January 1966.

6. Flynn, M. J., "Trends and Problems In Computer Organization," Information Processing 74, p. 3-10, 1974.

7. Knight, K. E., "Changes in Computer Performance," Datamation, V. 12, No. 9, p. 40-54, September 1966.

8. Raichelson, E. and Collins, G., "A Method for Comparing the Internal Operating Speeds of Computers," CACM, V. 7, No. 5, p. 309-310, May 1964.

9. Weitzman, C., Distributed Micro/Minicomputer Systems, p. 18-19, Prentice-Hall, Inc., 1980.

10. Honeywell Inc., St. Petersburgh Fla. Aerospace Div., ESD-TR-76-295, Secure Communications Processor Selection Trade Study, by C. H. Bonneau, March 1976.

11. Hodgins, B. D., "A Computer Evaluation Technique for Early Selection of Hardware" Naval Postgraduate School, December 1980.

12. Computer Science Department Stanford University Report CS-186, An Empirical Study of Fortran Program, by D. E. Knuth, p. 32, 1970.

13. Eckhouse, Jr., R. H. and Morris, L. R., Minicomputer Systems, 2nd ed., p. 138-139, p. 142-143, Prentice-Hall, 1979.

14. Barbour, R., Holcombe, J., Harris, C., and Moncrief, W., "Applications and Limitations of the Cost-Value Technique for Competitive Computer Selection" Proceedings, Computer Performance Evaluation Users Group 15th Meeting, 1979.

DEFICIENCES IN THE PROCESS OF PROCURING SMALL COMPUTER SYSTEMS: A CASE STUDY

Andrew C. Rucks

Peter M. Ginter

University of Arkansas
College of Business Administration
Fayetteville, AR 72701

The Latin phrase caveat emptor is nowhere more applicable than in the practice of acquiring computer systems for small business and small government entities. In a recent issue of Business Week, it was estimated that approximately 50 percent of the purchasers of small computer systems are dissatisfied with their systems. While many computer users are litigating vendors for redress of losses caused by insufficient data processing capability, this does not address the major failings of the process of procuring small systems. The small system procurement process produces a high rate of failure because many buyers inadequately analyze their data processing needs and fail to express these needs to potential vendors in a manner that will lead to the procurement of need satisfying computer systems.

Key words: Computer procurement; proposal evaluation; RFP preparation; small systems.

1. Introduction

Many small organizations have experienced dissatisfaction with procured computer systems [1]. Ample evidence of the frustration experienced by small organizations is found in recent litigations brought against computer vendors [2,3]. A recent procurement undertaken by the County Commission of Escambia County, Alabama offers a typical example of the major cause of user dissatisfaction -- a poorly developed solicitation document. The inadequacy of the Escambia County solicitation document was precipitated by a lack of technical expertise and a general reluctance to obtain assistance that could have contributed to the planning and execution of a successful procurement. The purpose of this paper is to (1) discuss a poorly prepared request for proposal (RFP), (2) indicate the probable consequences of issuing an inadequate request document and (3) offer a simple but effective RFP outline tailored for small organization procurements.

The RFP that is the basis for this paper was issued by the County Commission of Escambia County, Alabama in December 1980. Proposals were submitted to the Commission by three vendors. These vendors are not identified in this paper since to do so would be in variance with disclosure restrictions specified in the proposals.

2. Critique of the RFP

The Escambia County RFP is typical of proposals for which far too little data processing needs analysis has been performed and too little computer procurement expertise has been utilized. As a result, the RFP failed to precisely express the County Commission's requirements to potential vendors. The Escambia County RFP suffered from three major deficiencies: (1) performance specification vagueness, (2) requirements contradictions, and most importantly (3) a lack of functional requirements. The RFP was divided into seven major sections including: I--General Conditions, II--Vendor

Support, III--Escambia County's Responsibilities, IV--System Requirements, V--Software Requirements, VI--Form of Surety Guaranty, and VII--Appendix (see table 1). The major sections of the RFP will be reviewed briefly in order to delineate the proposals debilitating weaknesses.

Table 1. Contents of the Escambia County RFP

2.1 Section I--General Conditions

All three of the deficiencies suggested above were apparent in Section I of the Escambia County RFP. For instance, the RFP suggested "vendors may be required to demonstrate the capability of their proposed equipment and system software." Yet there was little in the RFP to suggest the actual capacity requirements. Also in this section, the RFP required vendors to "submit his best and cheapest proposal." This requirement was rather contradictory and provided little in the way of a meaningful statement of expectations. Additionally, the Escambia County RFP confounded vendors' proposals by demanding delivery of a "garbage collection service billing and collection system" but never itemized specifications in Section V--Software Requirements. The County also weakened considerably its negotiating position by indicating that it desired to consider only lease-purchase procurement plans. The RFP stated "Escambia County prefers a three-year, lease-purchase plan (lease with option to purchase)." This requirement precluded vendors from offering alternative methods of acquisition.

2.2 Section II--Vendor Support

The vendor support section of the Escambia County RFP likewise left considerable to the imagination in terms of specificity and clarity. The RFP called for "vendors... to supply both tested hardware and software" but failed to describe what tests were required. This section also contained such vagaries as: "the vendor is requested to furnish maintenance within four hours of being called on an 8-hour day basis..." and "The vendor would be required to train the county's personnel..."

2.3 Section IV--System Requirements

Section IV was potentially an important part of the RFP and yet this section contained specifications which were contradictory with the general conditions specified in Section I. Moreover Section IV contained statements such as "only equipment currently in production shall be proposed" which left the door open for a variety of vendor interpretations.

Within one subsection of Section IV was an attempt to specify the operating software for the system to be procured. The RFP speci-

fied a "disk-operating system." The meaning of this requirement was never clarified. Additionally, programming languages and utility software were treated with equally vague and non-specific language. For example the RFP stated:

> The vendor shall provide interpreters and/or compilers considered to be the prime programming language for the equipment offered. These programming languages shall be the latest and most up-to-date versions available. Also, sort and utility routines to support the system shall be supplied by the vendor.

2.4 Section V--Software Requirements

Applications software specifications were a consistent and persistent problem with the Escambia County RFP. Application software was mentioned in three sections of the RFP and these references were not consistent in terms of identifying the required application software. Table 2 illustrates these inconsistencies. It is interesting to note that even though "Voter Registration" software was mentioned in the Software Requirements section of the RFP, a statement of requirements for this software package was not included in the RFP. The property tax system was the only applications software item mentioned in all three applications software references.

Table 2. Application Software References in the Escambia County Alabama RFP

Application Software Title	RFP Paragraph References 1-10*	IV-1**	V***
Financial Management Information System	yes	no	yes
Property Tax Billing & Collection System	yes	yes	yes
Garbage Collection Service Billing & Collection System	yes	yes	no
Voter Registration	no	yes	yes
Automobile Liscense Records	no	yes	no

*Section title--Delivery and Installation
**Section title--System Summary
***Section title--Software Requirements

3.0 Vendor Responses

As a result of RFP specifications which were often contradictory and much too general, the task of evaluating the actual requirements of Escambia County was subject to a variety of interpretations. This vagueness provided vendors an opportunity to interpret the RFP according to their own equipment and software availability. In addition, vendor proposals could legitamately contain vagaries which seemingly satisfied the purchaser's needs. A summary of the vendor's responses is provided in table 3.

4. Proposed Remedies

The incompleteness, vagueness, and inconsistency of the Escambia County RFP precipitated the delivery of proposals that were incomplete and impossible to compare in terms of system costs. It was difficult to make a cost comparison because each of the vendors chose to interpret at least one of the "may include" clauses in the RFP to mean "not required", therefore, incongruous systems were proposed. As a result the Escambia County Commission was strongly urged to withdraw, rework, and resubmit its RFP.

4.1 RFP Outline for Small Organizations

An RFP becomes the basis for dealings between the organization acquiring a computer system and computer system vendors. Simply stated an RFP discloses to potential vendors thewho, what, where, when and how of a procurement action. Thus an RFP can be divided into as few as five major sections including: (1) procuring organization, (2) contract terms and conditions, (3) delivery schedule, (4) system specifications and, (5) operating environment.

4.1.1 Procuring Organization

This section of an RFP should be used to provide vendors with information concerning the business name and address of the organization issuing the RFP. The organization's agent(s), contracting officer, procurement manager and other responsible individuals and their responsibilities should be listed. It may be desirable to include a brief description of the mission of the organization and its history.

4.1.2 Contract terms and conditions

This section of the RFP delineates all procurement and contract administration procedures. Potential vendors must be told when proposals are due, the form of proposal sub-

Table 3. Summary of Vendor Responses

RFP Para.	Vendor I	Vendor II	Vendor III
I-1	Concurrance assumed	Concur	Unknown
I-2	Concurrance assumed	Concur	Unknown
I-3	Concurrance assumed	Concur	Unknown
I-4	Concurrance assumed	Concur	Unknown
I-5	Questionable concurrence	Concur	Unknown
I-6	Concurrance assumed	Concur	Concurrance assumed
I-7	Concurrance assumed	Exception: Escambia Co. to incur all "living/traveling cost" associated with a demonstration	Unknown
I-8	Concurrance assumed	Concur: includes copies of warranties and performance guarantee	Unknown
I-9	Qualified concurrance	Qualified concurrance	Unknown
I-10(a)	Concurrance assumed	Concur	Concurrance unknown, but assumed
(b)	Exception: 120 days after award (DAA)	Exception: 240-360 DAA	Unknown
(c)	Concurrance assumed	Exception: implementation by Oct. 1, 1981	Unknown
(d)	Concurrance assumed	Exception: implemented by Oct. 1, 1981	Unknown
(e)	Concurrance assumed	Concur	Unknown
I-11	Concurrance assumed	Concur	Unknown
I-12	Concur: No cancellation penalty at end of years 1 and 2 (unknown in third year); 50% lease payment credited to purchase, software not included in lease agreement	Concur: Cancellation penalty-the lesser of 20% of remaining contract value or 4 times the montly lease charge. No penalty charged if converted to purchase. Maximum price increase under contract 5% per year	Exception: does not offer a lease with option to purchase (LWOP) plan-offers "Government Installment Sales Plan" that calls for 10% down and the remainder financed over 36 or 60 months. Requires Escambia County to sign a "Non-Funding Clause" which disallows termination of the

Summary of Vendor Responses (cont.)

RFP Para.	Vendor I	Vendor II	Vendor III
			contract, except when the county does not have sufficient funds in a fiscal period to pay the contract amounts. The clause precludes Escambia County from securing replacement services, except from this vender curing the life of the contract.
I-13	Concurs with performance bond provision. Exception to requirement of specifications and vendor responses becoming part of contract.	Concurs with performance bond provision. Exception same as Vendor I	Unknown
II	Proven hardware and operating software supplied. An account manager will be assigned, the qualifications of this person were not supplied. The number and type of "Customer Service Representatives" assigned to the county is determined by the manager. Telephone support provided. No charge for support services. Concurrance assumed with provision of backup facilities. Page 3 of Appendix A indicates that Covington Co., is a comparable installation. However, does not indicate if backup can be obtained. Preventive maintenance performed monthly. On call maintenance with 4 hour response time between 8 and 5 (assumed to be 8 a.m. to 5 p.m.). Whether or not this applies 7 days per week is unknown. Parts depots in Montgomery and	Concur on tested hard-ware and software. A Marketing Representative and a Systems Engineer will be assigned as Project Manager(s). Some categories of System Engineer support offered at no charge, other types of service offered at the rate of $68.00 per hour. On call maintenance service available on a 24 hour per day, 7 day per work basis for major system components and 7 a.m. to 6 p.m. Monday through Friday for minor systems components. No added charge for leased systems, except for the minor system components serviced outside the afore mentioned limits which will be billed at a fee to be determined. No response time	Proven hardware and operating software will be supplied, the quality of application software is unknown. Does not address project manager or systems analyst support. No preventive maintenance offered. "Same day" response time offered, but no cost stated.

Summary of Vendor Responses (cont.)

RFP Para.	Vendor I	Vendor II	Vendor III
II (cont.)	Mobile. Concurs with training of country personnel.	guarantee is offered. Concurs with training needs.	
III	Concurrance assumed	Concurs	Unknown
IV-1	Concurrance assumed	Concurs	Concurrance assumed by virtue of a proposal being submitted.
IV-3	Master Control Program COBOL RPL	System Support Program RPGII compiler	Interactive Multi-programming Operating System COBOL Compiler
IV-4		No promised delivery schedule	Unknown
V-A 1.	General Ledger and Budgetary Accounting	Financial Management Information System	Financial Management Information System
V-A 2.	Payroll	Same as above	Same as above
V-A 3.	Same as V-A 1.	Same as above	Same as above
V-B	Property Tax System	Property (Real and Personal) Tax Billing and Collection System	Tax Collection System

mission, the procedure for contract award, the procedure for conducting face-to-face negotiations, the penalties to be imposed if the vendor fails to deliver any or all of the items specified in the RFP, the procedure to be followed for terminating a contract, and the procedure for resolving disputes arising from the contract. In summary the contract terms and conditions section of an RFP presents all of the legal understandings among the parties.

4.1.3 Delivery schedule

The delivery schedule should contain a list of all of the items to be delivered under the contract--not only hardware and software, but also training and manuals, etc.; and the date by which the items must be delivered. It is customary to express delivery dates in terms of days-after-contract award (DAA).

4.1.4 System specifications

This section informs potential vendors of the procuring organization needs in terms of a complete system. As a complete system specification, this section of an RFP is divided into five subsections: (1) hardware specifications, (2) software specifications, (3) system performance specifications, (4) maintenance requirements, and (5) training requirements.

Hardware specifications should be stated in functional terms. In other words, the procuring organization should not place itself in the position of specifying system architecture to vendors. Included in hardware specifications are the desired characteristics of terminals, printers, secondary storage devices, etc. Since it is unlikely that a live test demonstration (LTD) will be conducted as part of the pre-award proposal evaluation process, it is particularly important to be specific concerning the type and quantity of secondary storage.

Software specifications fall into two categories: system software and applications software. The system software to be delivered includes: an operating system, system utilities, text editor(s), programming

language interpreters and compilers, and programming support utilities (e.g. link/editor). For small systems, the description of these items should be general and functional in nature.

The major emphasis in developing the system specifications section of the RFP should be devoted to providing vendors with detailed descriptions of the applications software that is to be delivered. This description should include as much detail as possible concerning the purpose of the software, the quantity and description of input data, the quantity, frequency, and description of all reports to be generated by the software system.

The majority of procurements of large scale computer systems include provisions for an LTD. An LTD is used to determine if the data processing capacity of a proposed system is sufficient to meet the intermediate term (five to eight years) needs of the procuring organization. However, this is generally not feasible for small system acquisitions. The expense of preparing and conducting an LTD makes it impractical for small system procurements. The procuring organization may achieve some protection from insufficient capacity by requiring a "proof of capacity period" after delivery during which system capacity can be tested.

Maintenance and vendor support are perhaps the two most critical elements to be measured in the procurement of small computer systems. Since most small organizations do not have personnel with the technical background to repair either the software or hardware, the procuring organization must rely upon the vendor to provide these critical services. Therefore, it is important to precisely state requirements for maintenance of hardware and software. On-site maintenance, common for large installations, is impractical for small systems, therefore, the user of small systems must decide between on-call and per-call maintenance services. A good approach to this problem is to specify the temporal requirements for vendor response to service calls and then request that the vendor provide the organization with descriptions of their offerings in these areas.

Vendor training of the procuring organization's personnel must be specified in the RFP. The organization must decide the scope of training required and the location of training. The most frequently followed scenario is to identify the training required and have vendors provide a list of dates and locations for the desired training.

4.1.4 Operating Environment

This section discloses to vendors the characteristics of the site where the system will be installed. It should contain detailed drawings of the site with dimensions that will indicate to vendors such important considerations as the length of cabel runs. Other important information includes air conditioning and electrical system capacity.

References

[1] Computers: A burst of critical feedback, Business Week, February 2, 1981, pp. 68-71.

[2] Laberis, B., Judge OK's $2.3 million award to NCR user, Computerworld, May 25, 1981, p. 11.

[3] Davis, L., Burroughs admits to 160 suits, MIS Week, Vol. 2, No. 29, July 22, 1981, pp. 1+.

COST/PERFORMANCE COMPARISONS FOR STRUCTURAL ANALYSIS SOFTWARE ON MINI- AND MAINFRAME COMPUTERS

Anneliese K. von Mayrhauser
Raphael T. Haftka

Illinois Institute of Technology
Department of Computer Science
Chicago, IL 60616

Due to an ever-increasing selection of machines and widely different charging algorithms, the question often arises "on which machine does a program run most efficiently and/or cheaply". The program's behavior, i.e. its resource demands on the machine under consideration and their impacts on charges to be explored, as well as accuracy and correctness of the program's results and reliability of the computer service. Human factors influence decisions as well. We will discuss the selection of computer services from a user's point of view with an emphasis on structural analysis software. However, the data collection and evaluation procedure which analyses cost and performance relevant factors and their relationship with each other can also be applied to other software. The decision making process uses a set of automated tools which we developed.

Key words: Performance comparison; program behavior; structural analysis.

1. Introduction

A number of studies have described approaches for machine selection from a computer center management perspective ([1]-[4],[7]). Other work has taken the user point of view, but has only examined measurable technical factors such as reponse time and transmission rates ([8],[9]). In assessing the cost-effectiveness of computers or computer services for a particular application task, these methods turn out to be less than useful, because they do not deal with human factors adequately and they leave little room to compare different software packages which could be used to perform the application tasks with respect to their quality on the various machines under study.

Morgan and Campbell ([5]) suggest to construct an executable model of the set of programs to be run on machines under consideration because they are more flexible and avoid conversion problems. Since a user is also interested in the correctness of results of those programs on all machines, executable models of them will not do. Bell ([6]) reports that experimental comparisons are likely to be either very crude or very expensive. One of the biggest problems is that performance indices, resource consumption and therefore charges can vary for the same job, due to differing workload situations which the user is unable to control. This variability factor should be assessed and taken into account when selecting a computer service installation. Usually a more stable, predictable job cost is to be preferred- but this depends on the charging algorithm employed as well as on the machine characteristics. Because of these limitations as well as a limited budget for the study, accurate performance studies for some

problems were augmented by more informal approaches, such as a general experience with systems, studying installation statistics, or interviewing users, and limited model building for some programs.

One question which has been interesting the scientific computing community in the last few years is that of the cost-effectiveness and viability of minicomputers versus main frames. Minicomputers promise cheaper, more widely available computing facilities, but they pose many problems, particularly to those with large calculations in mind. The smaller main memory means that users have to make more use of disk I/O. Many minicomputers have a smaller word size with a devastating effect on accuracy. Storaasli and Foster ([10]) report 4-digit accuracy on PRIME 400 for a medium-sized problem as compared to 8-13 digit accuracy on a CDC CYBER 173. Longer elapsed times leave more room for machine hardware failures during a run. Pearson et al. ([11]) report some theoretical calculations required twenty-four hours or more, but the mean time between failures was also approximately twenty-four hours. Lack of adequate software libraries, debugging facilities, documentation and operations staff often hampers users of minicomputers. Some studies ([11] - [14]) show that based on machine cost alone minicomputers may be two to four times more cost-effective than main frames. The users of structural analysis software on minicomputers have conducted several studies that point to the economic advantages of using such machines (e.g., [10], [15]). These are based on data gathered by running structural analysis programs for a limited set of problems and machines.

When assessing the cost-effectiveness of services, the following steps should be followed:

(a) Set up of evaluation experiment.
This involves designing a representative set of problems which are to be solved by the computers. The software packages and the computers/computer services also need to be determined. These are the input variables. Output variables, e.g. the class of performance indices and other relevant information need to be determined as well. This will influence the relevant data to be collected during the measurement phase.

(b) benchmark and data collection and their analysis.
The set of problems is actually run on some of the machines and technical and human performance factors are observed. The raw performance data are processed and transformed into more concise information which enables their interpretation with respect to relevant performance indices.

(c) comparison, trade-off, evaluation
Finally the quality (or lack thereof) of the various choices is assessed with respect to all performance indices. This may be done by interpreting actual measurement data or by interpolation and/or extrapolation of measured data, or estimation of system performance through the use of results for another system.

As an example, let us take a user of structural analysis software. If he wants to assess the cost-effectiveness of a minicomputer versus a mainframe, the previously mentioned studies need to be extended in several directions. First, to use a wider mix of programs and more representative mix of problems for the comparison. Second, to include in the study several machines so that the results do not reflect only the charging algorithm at a particular installation. Last, to further assess the human factors involved in such comparison.

2. Experimental Design

Since the overwhelming majority of structural analysts use finite element methods, the present study is limited to such programs. By judicious choice of the type of problems and the type of structural analyses employed, one can obtain a reasonably reliable comparison. The following are the elements in such a comparison.

2.1 Most commonly used types of analysis

(i) Linear static solution for displacement and stresses
(ii) Linear eigenvalue analysis - calculation of buckling loads and vibration modes
(iii) Nonlinear response due to large deformations and material nonlinearities

2.2 Type of problem

(i) A simple cantilever beam
(ii) A plate with a hole
(iii) A stiffened cylinder

These problems are solved for the different analysis types. Several models are used for each problem, ranging from a very

crude model to a refined one, with varying degrees of freedom ranging from a few dozen for a crude model to more than a thousand for the refined one.

2.3 Computer programs

Three computer programs are used; these programs are:

1. SAP IV - A general purpose finite element program developed at the University of California at Berkeley is probably the most widely used "free" (it costs $200.) finite element code.

2. SPAR - A general purpose finite element program developed by W.D. Whetstone which serves as a prototype of a commercially developed code.

3. TWODEL - A special purpose finite element program developed by D. Malkus at IIT for large deformation analysis of two dimensional elasticity problems. It is representative of in house codes.

2.4 Computers and Charging algorithms

Our study on the cost-effectiveness of minicomputers and mainframes from a user's point of view uses the charging algorithms in several installations as the measure of the cost of running any of the selected problems. In most cases the charging algorithms are devised to recover the total cost of owning and operating the system, and in some cases to show profit.

Most of the runs were performed at the computer center at the Illinois Institute of Technology. The school operates its own PRIME 400 minicomputer and buys computing services on a UNIVAC 1100/81 from the United Computing System Corporation. A few runs were made also on the CDC CYBER 176 owned by a large manufacturing firm, and on the UCS CDC CYBER 176 and on the VAX 11/780 owned by the IIT Research Institute. The performance on these computers was used to predict costs on a few additional systems. The following is a brief description of the computer systems that were used in this study, their charging algorithms and the method used to predict costs on them.

IIT - PRIME 400 - The PRIME 400 is a medium size computer which is used at IIT to support interactive computing. The present configuration has 1.25 Megabytes of memory and two disk drives. The system is very busy from 9 AM Monday through Saturday with about 20 to 30 users. However, most of the users are not CPU intensive and do not put a heavy computing load on the system. As a result, the CPU intensive structural analysis programs often show very good response times even during the day. In running on the PRIME 400 it was found that the response time and I/O times are quite sensitive to the workload while the CPU time is not. The amount of workload depends on the number of users, but more importantly on the kind of work they are doing, i.e. whether they are doing simple editing tasks or compiling and running CPU intensive jobs with considerable working set size or core requirements. To account for the variability, runs were performed for light, medium, and heavy load periods on the PRIME. The results were averaged to indicate an expected result, as well as to give an idea of the possible deviation from it. It is important to realize that a considerable spread in the consumption of a particular reource need not entail the same difference in cost. Whether it does or not depends on how sensitive the corresponding charging algorithm is with respect to that response. The charging algorithm for the system is given in Table 1 (considerable discounts are available at non-business hours).

NASA - Langley Research Center PRIME 400- The NASA Langley Computer Center operates several minicomputers including a PRIME 400 computer. The charging algorithm (see Table 1) was devised to recover the capital and operating costs. Although the NASA PRIME 400 supports a small number of users, they run more CPU intensive jobs than the IIT users resulting in a comparable system load. Therefore it was assumed that program performance is similar on the IIT and NASA systems. Experience confirmed this. The cost of running a job on the NASA PRIME 400 was calculated based on this assumption. The two PRIME computer services charge for an unstable resource such as wall time (one may also call it connect time, since these services are interactive); this was not uncommon among the charging algorithms for computer services which we encountered. The charge serves a regulatory function, because it is a deterrent when the system is crowded and any additional work would slow it down further. For long jobs it also encourages the use of batch runs when they are offered as an alternative.

NASA - Langley Research Center Cyber 173 The NASA Langley Research Center operates several CDC main frames including two CYBER 173 computers which are used to support interactive computation. The charging algorithm is devised to recover capital and operating expenses.

Haftka has an extensive experience using the SPAR program on both the IIT and NASA SYSTEMS. Based on this experience the following assumptions were used to predict cost on the NASA Cyber 173.

1. The CPU time on the Cyber is 1.5 times that of the CPU time on the UNIVAC 1100/81.

2. The I/O charges are calculated based on the number of reads and writes from the SPAR data base available in the SPAR output. It was assumed that these numbers are similar for both systems, and that the charge per disk access is 1.1 times the minimum charge per access. This is based on the feature of the charging algorithm which is relatively insensitive to the number of words in each read or write operation.

3. The amount of core required is assumed to vary from 70K octal for the smallest number of nodes to 120K for the largest number of nodes.

4. Based on the above considerations the cost of running SPAR on the NASA Cyber is:

$$\text{Cost} = (1 + 0.048A)\ (0.0136T_1 + 0.003ni\emptyset)$$

where

T_1 = UNIVAC 1108/81 CPU time (sec)

$ni\emptyset$ = combined number of disk reads and writes reported by SPAR

A = core storage (in units of 10K octal words)

Large Manufacturing Company CDC Cyber 176 - A few runs of SPAR were made on a Cyber 176 which is owned by a large manufacturing company. The charging algorithm is given in Table 1. It appears to be fairly high on CPU and low on I/O and core storage charges. In predicting costs on this machine it was assumed that CPU times can be predicted from the UNIVAC 1100/81 results based on published data ([16]) rating the UNIVAC 1100/81 at 1800 KOPS and the Cyber 176 at 9300 KOPS. It was assumed that the ratio of I/O time to CPU time is the same on both machines. Because the I/O cost on the Cyber 176 is low, even a large error in this assumption is not expected to change the cost much.

Concordia College Computer Network (CCCN) CCCN provides administrative, instructional and research computing services to a large number of educational institutions. The service includes a large software development staff and the charges are computed to recover costs of operating the center. Interactive and batch services are provided by a UNIVAC 90/80 mainframe installed in December 1980. The UNIVAC 90/80 is roughly equivalent to an IBM 370/158 in raw processing power.

The prediction of costs of running on the CCCN UNIVAC 90/80 were based on the charging algorithm given in Table 1 with the following assumptions:

1. CPU times are 2.25 longer than on the UNIVAC 1100/81. This is based on published data ([16]) rating the UNIVAC 1100/81 at 1800 KOPS and the UNIVAC 80/90 at 800 KOPS.

2. Core requirements were estimated at 200K bytes for the small problems and at 300K bytes for the large problems.

IIT Research Institute VAX 11/780 - The IIT Research Institute (IITRI) operates a DEC VAX 11/780 with 3 Megabytes of core memory and 650 Megabyte disk space. It does not have a floating point processor. As it is not heavily loaded, time is available to outside users. The charges are given in Table 1.

2.5 Performance indices

As part of the design, performance variables must be chosen. The "utility" of a service to a user is a function of those performance factors which affect the value and usefulness to the user and thus necessarily subjective and dependent on particular needs in a given situation. For this study we chose the following performance variables as being most important in comparing the computer service alternative.

1. correctness of results

2. turnaround time

3. system availability or "uptime"

4. support - technical help, documentation

These factors are fairly self explanatory and were selected because they are obviously important. Other situations and users may choose a different set of factors which reflects their goals better. For a comprehensive list of performance variables see

[17] and [18].

3. Benchmark and data collection analysis

3.1 Data collection and analysis

Measurement data collected during the benchmark are used for job cost prediction and performance comparison of the two machines. The following statistics are collected for each run:

- -CPU time (measured in seconds)
- -I/O time (measured in seconds)
- -memory used (measured in K bytes)
- -turnaround time (measured in seconds)
- -number of disk I/O operations
- -job cost (measured in $)

A comprehensive program performance measurement system was developed for the data collection and analysis. This system will automatically instrument the FORTRAN source programs by inserting CALL statements at the entry and exit points of each module in the system. The CALL statements invoke a routine called REFSTR which will collect performance data. Example:

```
      SUBROUTINE ELT3A4 (A,B)
      CALL REFSTR ('ELT3A4', 'ENTRY')
      .
      . body of subroutine
      .
      IF (ISTOP.EQ.0) GOTO 99999
      .
      .
99999 CALL REFSTR ('ELT3A4', 'EXIT')
      RETURN
      END
```

The REFSTR data collection routine logs entries and exits from modules in the system, captures the CPU and I/O times at each of these points, and writes this program trace information out to a permanent file in very large blocks. The program trace data from REFSTR looks like this:

ROUTINE	CPU	I/O
$MAIN$ ENTRY	11.1694	9.1964
FACMGT ENTRY	11.1704	9.1964
FACMGT EXIT	13.0202	9.1964
LODCMN ENTRY	13.0246	9.1992
INPUTJ ENTRY	13.0522	9.2172
ELTYPE ENTRY	13.4028	9.4352
PLANE ENTRY	13.4034	9.4352
ELT3A4 ENTRY	13.4098	9.4454
PLNAX ENTRY	13.4104	9.4454

Calibration studies have been performed on REFSTR to determine the CPU and I/O overhead which is generated by the data collection itself. This instrumentation overhead is removed during analysis by the ANALYZ program. ANALYZ processes the raw data from REFSTR, removes instrumentation overhead, and prints summary performance results (Figure 1). ANALYZ also produces the following dynamic call matrix to be used for further analysis or program restructuring for virtual memory systems (Figure 2). A routine called LSP was developed to fit polynomial curves to the resource consumption and cost data using a least squares technique. LSP also can plot the data points and fitted curves.

The benchmark performance data is collected at the job level for all jobs, at the program level for all SPAR jobs, and at the subroutine level for a few selected SAPIV jobs. Since SPAR is a modular sequence of rather independent tasks, it was not necessary to collect measurement data at the subroutine level. SAPIV, on the other hand, is not sequentially structured, but is a conglomerate of subroutines with frequent interaction. This necessitates more detailed performance data collection from which SAPIV's resource consumption models are derived (see sample output from ANALYZ for example of detailed analysis). The resource consumption statistics are collected differently on each machine. In each case instrumentation overhead is measured and removed from the data, if it is significant. Job cost computations are based on resource consumption and are not directly taken from the job accounting output.

3.2 Benchmarks and their results

3.2.1 Beam problem

A cantilever beam was modeled by plane beam elements (E24 elements in SPAR) and three vibration modes and frequencies were calculated using the SPAR program. The number of nodes was varied from 5 to 600 degrees of freedom per node. This simple problem exposed a bug in the PRIME version of SPAR. The program could not calculate more than two vibration modes and frequencies. On the UNIVAC there was no problem. Performance data and cost (prime time rate) for the beam problems are given in Tables 2 (UNIVAC 1100/81) and 3 (PRIME 400). The total cost proved to be very close to a linear function of the number of nodes for both computers. It was therefore possible to predict accurately the cost of 600 node runs by extrapolating the cost of the 5-280 node runs, It is therefore assumed that the 5-600 node results can be used safely to extrapolate cost up to 1200 nodes. Figure 3 shows the actual cost points

and the predicted cost curves for both computers. The triangles represent cost points for the UNIVAC 1100/81, the circles averages for different load situations on the PRIME 400. It can be seen that although the results may have been sensitive for certain resources, they are not very sensitive in their cost, and cost predictions for other problem sizes can be expected to lie within an analogous range.

The beam runs are about twice as expensive to run on the UCS UNIVAC 1100/81 than on the IIT PRIME 400. The response time or turnaround time is favorable on the PRIME for the small problems. The UNIVAC is quite faster for the larger problems.

The performance of the other computers for the beam problem was actually measured or estimated and the results are summarized in Table 4. The costs of running the problen on the NASA PRIME-400 are comparable to those at IIT. The UCS UNIVAC 1100/81 is the most expensive mainframe. The low cost on the Cyber 176 is a result of low I/O charges on that machine compared to the NASA Cyber 173 ($0.84 of $2.25 versus $8.00 of $ 8.60). When evaluating the total cost results in Table 4 the main difference is not between minicomputers and mainframes but between service bureau computers (the two UNIVAC machines) and the user owned machines and their lower charges.

3.2.2 Plate problem

A rectangular plate with a hole was modeled with quadrilateral plane stress finite elements (Figure 4 insert). It is subjected to uniform loads. Stresses are calculated with the number of nodes varying between 35 and 594. The problem was run with the SAP IV and SPAR programs. Tables 5 and 6 show results for various machines.

The turnaround time on the UNIVAC is much better than that of the PRIME. Just as for the beam problems, the measured data were used to predict costs of running problems upto 1200 nodes. The cost data and the predictive curve are given in Figures 4 (SPAR) and 5 (SAPIV). The cost of the plate problems increases much more rapidly on the PRIME than on the UNIVAC. Table 7 shows a cost comparison for all computers based on the data in Table 5 and 6. For SAPIV, the user owned CDC has the lowest charges whereas the two UNIVACs and the minicomputer IIT PRIME 400 are most expensive. For SPAR, the 1200 node figures show a clear disadvantage for the IIT PRIME 400.

The plate problem was now analyzed for large deformations using the TWODEL program. Performance and cost variables for the UCS UNIVAC 1100/81 and the PRIME 400 are given in Tables 8 and 9. The problem is much cheaper to run on the PRIME 400 than on the UNIVAC 1100/81. The 289 node problem was the largest that could be run on the UNIVAC 1100/81 because of core limitations. The PRIME 400 with its virtual memory could handle larger problems. The excellent performance of the program on the PRIME 400 is attributed to two factors. First, the large core requirements on the UNIVAC 1100/81 are very expensive (fifty percent of the cost of the 289 run was core cost). Second, the program was written by the developer on both machines. As a result it does not suffer from the deterioration in performance that afflicts programs that have been converted to different machines by users who are not as knowledgeable about the program as its developer. This applies to the SPAR and SAP IV programs. A comparison of the CPU times on the PRIME 400 and the UNIVAC 1100/81 for the plate problem shows the following ratios. The PRIME 400 CPU times are 7-15 times longer for SAP IV, 7-13 times longer for SPAR, but only 5-8 times longer for TWODEL. The comparison with other computers is given in Table 10. The cost on the two UNIVAC computers is very high and the cost on the CDC computer is very low compared to the PRIME 400.

3.2.3 Stiffened cylinder problem

A stiffened cylindrical shell, see figure 6, modeled by plate and beam elements. The lowest four vibration frequencies and the buckling load of the cylinder were calculated using the SPAR program. Three models were used with 5x5, 10x10, 15x15 grids of elements or 36, 121 and 256 nodes, respectively.

The cylinder problem revealed two bugs in the SPAR program on the PRIME. The stress analysis which is preliminary to the buckling calculation was not correct when combined membrane-bending elements were employed. As a result it was necessary to use twice as many plate elements; pure membrane elements and pure bending elements. Unfortunately, the mass matrix was not calculated properly with this replacement. As a result we had to perform the calculations in two separate runs, one using double elements for buckling analysis and one using combined elements for the vibration analysis. This did not increase the cost of the run substantially but was an inconvenience for the analyst.

The results of CPU and I/O times, reponse time and cost are given in Table 11 for the UCS UNIVAC 1100/81 and in Table 12 for the IIT PRIME 400. Comparison with other computers is given in Table 13 for the

largest problem of 256 nodes. It can be seen that for this problem the performance of the PRIME 400 is dismal and that of the CDC very good. Next the same problem was run with the SAP IV program and produced quite different results. The PRIME 400 performed excellently whereas the UNIVAC 1100/81 and the UCS Cyber 176 had convergence problems. A change of parameters solved them for the 36 node problem, but difficulties remained for larger ones. To resolve these was beyond the knowledge of an average user.

The benchmark results showed that correctness of the results cannot automatically be assumed. One problem area was improper conversion for those packages which were developed on one machine and subsequently transported to another. We found several problem types that could not be run correctly on the PRIME due to those problems (30 percent of the beam runs and 50 percent of the plate runs were completely correct when run on the PRIME). Another contributing factor lies in the fact that certain problems require high precision floating point values to be stored and manipulated and the minicomputer's smaller word size and floating point operand size can produce incorrect or poor results. On the other hand, some of the mainframes exhibited correctness and feasibility problems as well. The most important thing a user can do to protect himself is to check results carefully and validate them, if possible.

3.3 Other performance factors

Availability was considered important. Table 14 shows percentage of uptime, mean time between failures (MBTF) and mean time to repair (MTTR) for UCS UNIVAC 1100/81 and PRIME 400. The UNIVAC data is taken from published figures (UCS "Data Link") for January through December 1980. PRIME 400 data could only be obtained for the months of October through December 1980. The PRIME system is down fewer times than the UNIVAC, but is down for extended periods of time. The overall uptime of the UNIVAC is somewhat better than the PRIME.

Support is a very important performance factor for many users. Support includes availability and quality of technical help and documentation. We found both the UNIVAC and PRIME centers to be somewhat deficient in support. We found the UNIVAC documentation to be fairly good, the access to documentation to be good, but the access to technical help to be poor (due to contract between service bureau and university). We found the PRIME documentation to be weak and unavailable for reference on site. Purchase is always required. Only standard PRIME manuals are available. Information about IIT PRIME 400 particularities is communicated by "word of mouth". Although the technical people are fairly competent, they are far overburdened and unable to provide a consistent level of support.

The results for all performance factors can be used with a weighted scoring method to determine which of the machines has the highest utility and to perform a utility/cost trade off analysis. For a subset of the results in this paper such an analysis was performed in [17], the same method could be used for the results presented here.

4 Concluding remarks

The study of the performance of structural analysis programs on mainframe and minicomputers has not shown a clear cut advantage of either type of computer. Both mainframe and minicomputer had difficulties, but with different problems. For minicomputers some of these problems were due to portability problems (SAP IV, SPAR) and imperfect conversion from mainframe software to a minicomputer. As the problem sizes increased, the minicomputers were less efficient for those programs which were not developed for them; but some of the mainframes could not run those problems adequately, either. The minicomputer clearly outperformed the main frame for the one program which was developed on the minicomputer (TWODEL).

The cost comparison does not show any drastic difference between main frames and minicomputers except for the user owned CDC CYBER. This probably reflects a competitive marketplace where the decrease in the cost of raw computation has stimulated an increase in programming and computer services offered. There is no longer a main frame monopoly on (structural analysis) software and services have to be priced to be competitive with minicomputers. The user owned machines which just charge to cover their costs naturally are cheaper than those which need to show profit and may have marketing and other related expenses. This reduces the problem to one of microeconomics and price theory ([19]).

The study reported in this paper has been supported by the office of Naval Research under contract No. N00014-80-C-0364. The authors wish to acknowledge the help of Mr. Dennis Witte in obtaining and processing the performance data.

References

[1] Randal, J.M., and Badger, G.F., Using Quantitative Criteria for Computer Selection , Computing Services Office, University of Illinois at Urbana-Champaign, Urbana, IL, 1976.

[2] Sharpe, W.F., The Economics of Computers, Columbia University Press, New York, 1969.

[3] Strauss, J.C., A Benchmark Study, AFIPS Conference Proceedings 41, FJCC, 1972, pp. 1225-1233.

[4] Kernighan, B.W., Plauger, P.J., and Plauger, D.J., On Comparing Apples and Oranges, or why my Machine is better than your Machine, Performance Evaluation Review, Vol. 1, No. 3, 1972, pp. 16-20.

[5] Morgan, D.E. and Campbell, J.A., An Answer to a User's Plea?, Proc. 1st ACM-SIGME Symp. on Measurement and Evaluation, February 1973, pp. 112-120.

[6] Bell, T.E., Computer Performance Variability, AFIPS Conf. Proc. 43, NCC, 1974, pp. 761-766.

[7] Timmreck, E.M., Computer Selection Methodology, Computing Surveys, Vol. 5, No. 4, 1973, pp. 199-222.

[8] Mamrak, S.A. and Amer, P.D., Comparing Interactive Computer Services - Theoretical, Technical, and Economic Feasibility, Proc. NCC, 1979, pp. 781-787.

[9] Mamrak, S.A. and DeRuyter, P.A., Statistical Methods for the Comparison of Computer Services, Computer, Vol. 10, No. 11, 1977, pp. 32-39.

[10] Storaasli, O.O. and Foster, E.P., Cost-Effective Use of Minicomputers to Solve Structural Problems, AIAA Paper No. 78-484, 1978.

[11] Pearson, P.K., Luchese, R.R., Miller, W.H., and Schaefer, H.F., Theoretical Chemistry Via Minicomputer, Minicomputers and Large Scale Computations, American Chemical Society, 1977, pp. 171-190.

[12] Norbeck, J.M. and Certain, P.R., Large Scale Computation on a Virtual Memory Minicomputer, Minicomputers and Large Scale Computations, American Chemical Society, 1977, pp. 191-199.

[13] Wagner, A.F. Day, P., Van Buskirk, R., and Wahl, A.C., Computation in Quantum Chemistry on a Multi-Experiment Control and Data-Acquisition on a Sigma 5 Minicomputer, Minicomputers and Large Scale Computations, American Chemical Society, 1977, pp. 200-217.

[14] Freuler, R.J. and Petrie, S.L., An Effective Mix of Minicomputer Power and Large Scale Computers for Complex Fluid Mechanical Calculations, Minicomputers and Large Scale Computations, American Chemical Society, 1977, pp. 218-234.

[15] Conway, J.H., The Economics of Structural Analysis on Superminis, Proc. 7th ASCE Conference on Electronic Computation, 1979, pp. 373-385.

[16] Lias, E.J., Tracking the Elusive KOPS, Datamation, November 1980, pp. 99-105.

[17] Von Mayrhauser, A.K. and Witte, D.E., Cost/Performance Comparisons for Programs on Different Machines, to be presented at ECOMA.9, October 6-9, 1981, Copenhagen.

[18] Miller, W.G., Selection Criteria for Computer System Adoption, Educational Technology, Vol. 9, No. 10, 1969, pp. 71-75.

[19] Cotton, I.W., Microeconomics and the Market for Computer Services, Computing Services, Vol. 7, No. 2, 1975, pp. 95-111.

Table 1. Charging algorithms (dollars)

1. IIT PRIME 400	$0.02 (T_1 + T_2) + T_3$ T_1 = CPU time (sec) T_2 = I/O time (sec) T_3 = wall time (hours)
2. NASA Langley Research Center - PRIME 400	$30T_1 + 15T_2$ T_1 = CPU time (hours) T_2 = wall time (hours)
3. UCS UNIVAC 1100/81	$0.18T_1 + 0.0011A(T_1 + T_2)$ T_1 = CPU time (sec) T_2 = I/O time (sec) A = (core/512) words
4. Concordia College UNIVAC 90/80	$(0.156 + 0.0000248A)T$ T = CPU time (sec) A = core (kilobytes)
5. Large Manufacturing Company CDC CYBER 176	$0.2T_1 = 0.0003A (T_1 + T_2) + .03 * T_2$ T_1 = CPU time (sec) T_2 = I/O time (sec) A = core (kilowords)
6. NASA Langley Research Center - CDC CYBER 173	See section on "Computers and Charging Algorithms," pages 2-4.
7. UCS CDC CYBER 176	Confidential
8. IIT Research Institute VAX 11/80	$0.05T_1 + 5T_2$ T_1 = CPU time (sec) T_2 = connect time (sec)

Table 2. Beam results on the UCS UNIVAC 1100/81 using SPAR

Number of Nodes	CPU (sec)	I/O Time (sec)	Turnaround Time (min)	Cost (dollars)
5	5.06	15.77	3:21	1.83
25	5.74	16.12	2:50	2.07
60	7.07	18.17	6:04	2.66
90	8.30	19.79	2:41	3.16
125	10.29	22.81	4:26	4.07
280	16.70	32.59	7:17	6.90
450	22.52	41.73	3:16	9.53
600	30.73	59.68	7:45	13.80
1200 *	60.76	72.12	8:55	30.89

* predicted

Table 3. Beam results on the IIT PRIME 400 (averages)

Number of Nodes	CPU Time (sec)	I/O Time (sec)	Response Time	Cost (dollars)
5	14.70	17.68	1:03	0.67
25	19.33	17.43	1:03	0.73
60	30.65	28.45	2:37	1.22
90	37.07	22.84	1:57	1.23
125	47.82	37.25	6:02	1.80
280	94.24	63.11	5:41	3.25
450	141.07	132.87	11:04	5.16
600	197.38	125.76	13:47	6.68
1200 *	422.92	262.02	26:36	14.14

* predicted

Table 4. Comparison of the 600 and 1200 node beam problems on various computers (the estimated 1200 node results are given in parentheses)

Computer	CPU (sec)	I/O (sec)	Turnaround Time (min)	Total Cost (dollars)
IIT PRIME 400+	197.38 (422.92)	125.76 (262.02)	13:47 (26:36)	6.68 (14.14)
NASA LRC PRIME 400*	197.38 (422.92)	125.76 (262.02)	13:47 (26:36)	5.08 (10.17)
UCS UNIVAC 1100/81	30.73 (60.76)	59.68 (72.10)	7:45 (8:55)	13.80 (30.89)
NASA LRC CYBER 173	46.1 (91.14)			8.60 (20.78)
LMC CYBER 176	6.05 (11.76)	24.5 (35.4)		2.24 (3.98)
Concordia College UNIVAC 90/80	69.14 (136.71)			11.13 (22.34)

* estimated
+ average

Table 5. Plate linear analysis with SPAR on various computers

Number of Nodes	CPU (sec)	I/O (sec)	Turnaround (min:sec)	Cost (dollars)
UNIVAC 1100/81				
35	4.80	14.25	1:27	1.69
72	6.98	15.55	1:44	2.46
143	11.89	18.36	2:08	4.22
285	24.46	25.27	2.48	8.69
594	60.58	44.78	4:36	21.94
1200	170.27	102.31	9:23	62.89
IIT PRIME 400				
35	32.11	19.18	1:23	1.05
72	58.40	29.91	2:58	1.81
143	123.14	44.96	4:30	4.18
285	285.14	123.65	8:56	8.39
594	805.96	679.25	41:24	28.71
1200	2500.40	1370.10	144:15	126.21

Table 6. Plate linear analysis with SAP IV on various computers

Number of Nodes	CPU (sec)	I/O (sec)	Turnaround (min:sec)	Cost (dollars)
UNIVAC 1100/81				
35	4.52	1.11	1:13	0.90
72	6.77	1.93	1:51	1.56
143	11.85	3.54	2:12	2.97
285	25.38	8.19	3:08	7.15
594	78.59	26.94	10.08	23.61
1200*	316.58	122.14	18:54	98.67
IIT PRIME 400+				
35	30.82	25.65	7:55	1.26
72	66.40	41.15	11:00	2.33
143	166.12	47.15	21:27	4.62
285	396.80	82.61	25:32	10.01
594	1186.52	216.10	54:08	28.95
1200*	3791.00	685.74	112:12	108.60
UCS CYBER 176				
35	.54	1.88	--	0.96
72	1.00	3.08	--	1.92
143	1.69	4.48	--	3.36
285	4.10	10.82	--	7.20
594	11.96	38.21	--	21.60
1200*	32.96	99.18	--	70.81
IIT Research Institute VAX 11/780 +,**				
35	10.36	--	0:25	0.52
72	20.94	--	0:30	1.05
143	59.62	--	3:30	2.98
285	182.43	--	5:28	9.12
594	568.33	--	12:41	28.42

* estimates
\+ averages
** jobs were run in batch mode, no connect time charges

Table 7. Comparison of the cost of linear analysis of the 594 (1200) node plate model on various computers (in dollars)

Computer System	SAP IV 594	SAP IV 1200*	SPAR 594	SPAR 1200*
IIT PRIME 400+	28.95	108.65	28.71	126.21
NASA LRC PRIME 400*	23.38	59.64	17.06	56.90
UCS UNIVAC 1100/81	23.61	98.67	21.94	62.89
NASA LRC CYBER 173	**		6.84	18.90
LMC CYBER 176*	4.23	11.39	2.85	7.82
Concordia College UNIVAC 90/80	28.98	116.42	22.28	69.57
UCS CYBER 176	21.60	70.81	--	--
IIT Research Institute VAX 11/780	28.42	**	--	--

\+ average of several runs
* predicted results
** data insufficient for prediction

Table 8. Plate nonlinear analysis on the UCS UNIVAC 1100/81

Number of Nodes	CPU Time (sec)	I/O Time (sec)	Core Storage (Kwords)	Turnaround Time (min)	Cost (dollars)
35	19.97	0.35	22.9	32:47	4.61
72	54.15	0.66	28.4	38:01	13.13
143	156.91	1.29	42.5	55:10	42.68
289	549.69	1.78	76.0	137:21	189.33
600*	2030.59	3.59	175.7	650:43	1151.80

* estimates

Table 9. Plate nonlinear analysis on the IIT PRIME 400

Number of Nodes	CPU Time	I/O Time (sec)	Response Time (min)	Cost (dollars)
35	156.01	2.74	7:08	3.29
72	379.95	3.81	10:09	7.84
143	978.05	19.47	29:59	20.45
289	2693.59	69.70	70:50	56.44
600*	8693.00	287.91	204:14	193.06

* estimates

Table 10. Comparison of the cost (in dollars) of nonlinear analysis of the 289 and 600 node plate models on various computers using the TWODEL program

Computer System	CPU Time (sec) 289	CPU Time (sec) 600*	Core+ 289	Core+ 600*	Cost (dollars) 289	Cost (dollars) 600*
IIT PRIME 400	2694.	8693.	--	--	56.44	193.06
NASA LRC PRIME 400*	2694.	8693.	--	--	40.13	123.50
UCS UNIVAC 1100/81	550.	2030.6	76.0	175.7	189.33	1177.66
LMC CDC CYBER 176*	58.8	217.1	42.2	97.6	12.52	49.80
Concordia College UNIVAC*	1238.	4570.7	320.	740.	203.03	794.20

* estimates: + UCS UNIVAC 1100/81 and LMC CDC CYBER 176 core requirements are given in K words. Concordia College UNIVAC core requirements are given in K bytes.

Table 11. Cylinder buckling and vibration analysis on the UCS UNIVAC 1100/81

Number of Nodes	CPU Time (sec)	I/O Time (sec)	Turnaround Time (min)	Cost (dollars)
36	27.28	19.34	7:09	7.36
121	110.65	28.16	10:11	29.21
256	302.59	48.45	19:52	79.97

Table 12. Cylinder buckling and vibration analysis on the IIT PRIME 400

Number of Nodes	CPU Time (sec)	I/O Time (sec)	Turnaround Time (min)	Cost (dollars)
36	579.93	343.28	64	19.53
121	3197.10	1184.54	156	90.23
256	10784.0	6589.98	774	360.38

Table 13. Comparison of the 256 cylinder problem on various computers

Computer	CPU Time (sec)	I/O Time (sec)	Total Cost (dollars)
IIT PRIME 400+	10784	6590	360.38
NASA LRC PRIME-400*	10784	6590	283.37
UCS UNIVAC 1100/81	302.59	48.45	79.97
Concordia College UNIVAC 90/80*	680.8	--	110.96
LMC CDC CYBER 176*	58.2	24.2	12.64
NASA LRC CYBER 173*	453.9	--	29.55

+ averages

* predicted results

Table 14. Uptime - downtime characteristics

	UCS UNIVAC 1100/81	IIT PRIME 400
Uptime %	98.9	96.7
MBTF (hours)	56.79	87.98
MTTR (minutes)	34.7	178.2

MODULE	CPU	%CPU	I/O	%I/O	#CALLS	#RETURNS
$MAIN$	0.065	.08	.007	.04	1	1
FACMGT	1.849	2.38	.0	.0	1	1
ELAW	0.139	.18	.128	.48	546	546
ADDSTF	21.396	27.51	9.023	34.09	1	1
.						
.						
.						
SESOL	31.583	40.61	10.213	38.59	1	1
STRSC	.944	1.21	1.255	4.74	1092	1092
TOTAL	77.769		26.467			

CPU = amount of CPU time spent in module
%CPU = fraction of total CPU time spent in module
I/O = amount of I/O time spent in module
%I/O = fraction of total I/O time spent in module
#CALLS = number of entries into module
#RETURNS = number of exits from module

Figure 1. Summary performance results

CALL MATRIX

TO:	1	2	3	4	5	6	7	8	9	10
FROM:										
1 $MAIN$	0	1	1	1	1	0	0	0	0	0
2 FACMGT	0	0	0	0	0	0	0	0	0	0
3 LODCMN	0	0	0	0	0	0	0	0	0	0
4 INPUTJ	0	0	0	0	0	0	0	0	0	0
5 ELTYPE	0	0	0	0	0	2	0	0	0	0
6 PLANE	0	0	0	0	0	0	1	0	0	0
7 ELT3A4	0	0	0	0	0	0	0	1	0	0
8 PLNAX	0	0	0	0	0	0	0	0	546	0
9 ELAW	0	0	0	0	0	0	0	0	0	546
10 POSINV	0	0	0	0	0	0	0	0	0	0
.										
.										
.										

Figure 2. Dynamic call matrix

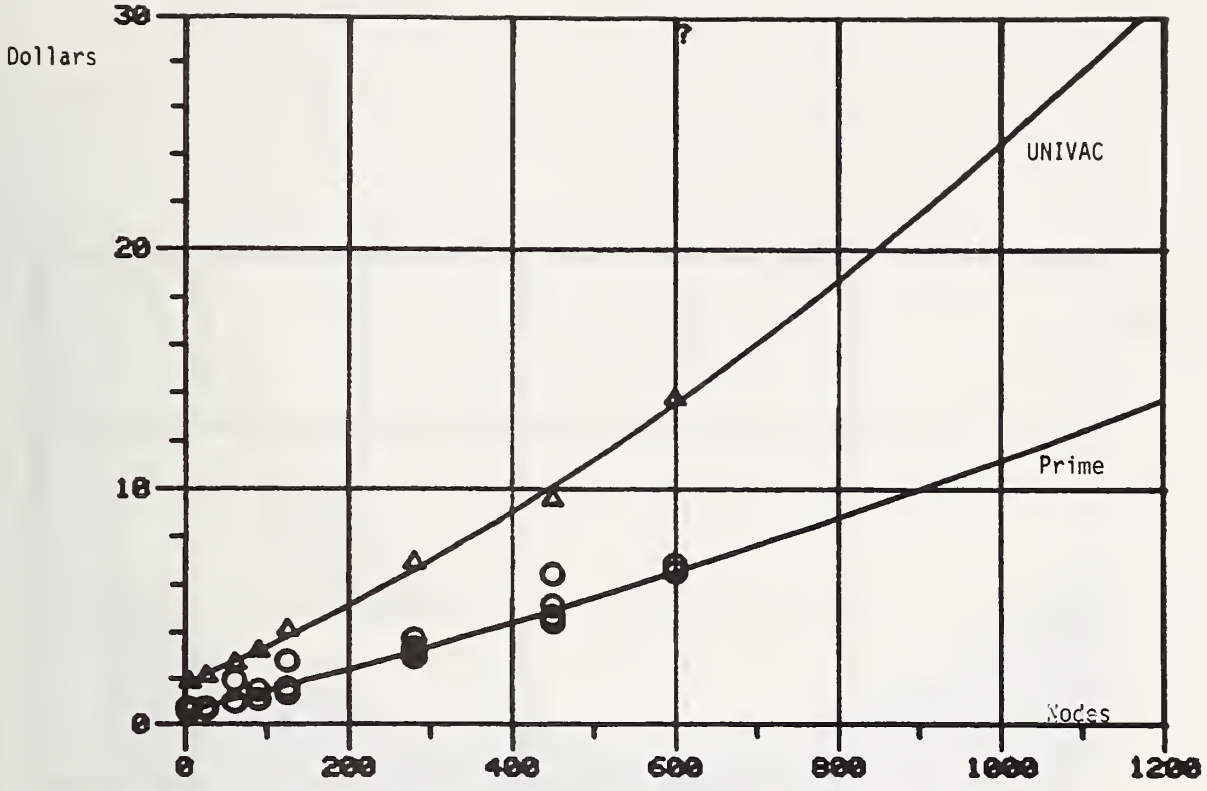


Figure 3. Total cost of running beam problem on UCS UNIVAC 1100/81 and PRIME 400

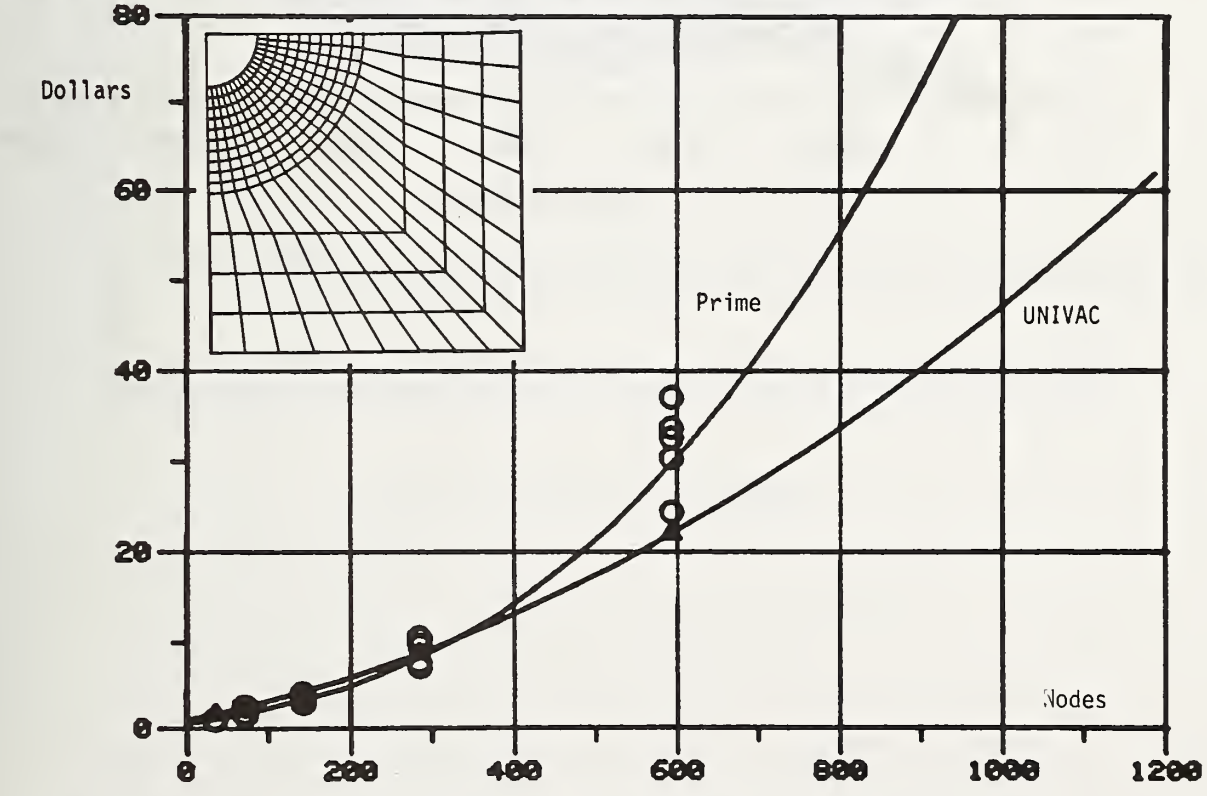


Figure 4. Comparison of total cost for the plate problem using the SPAR program

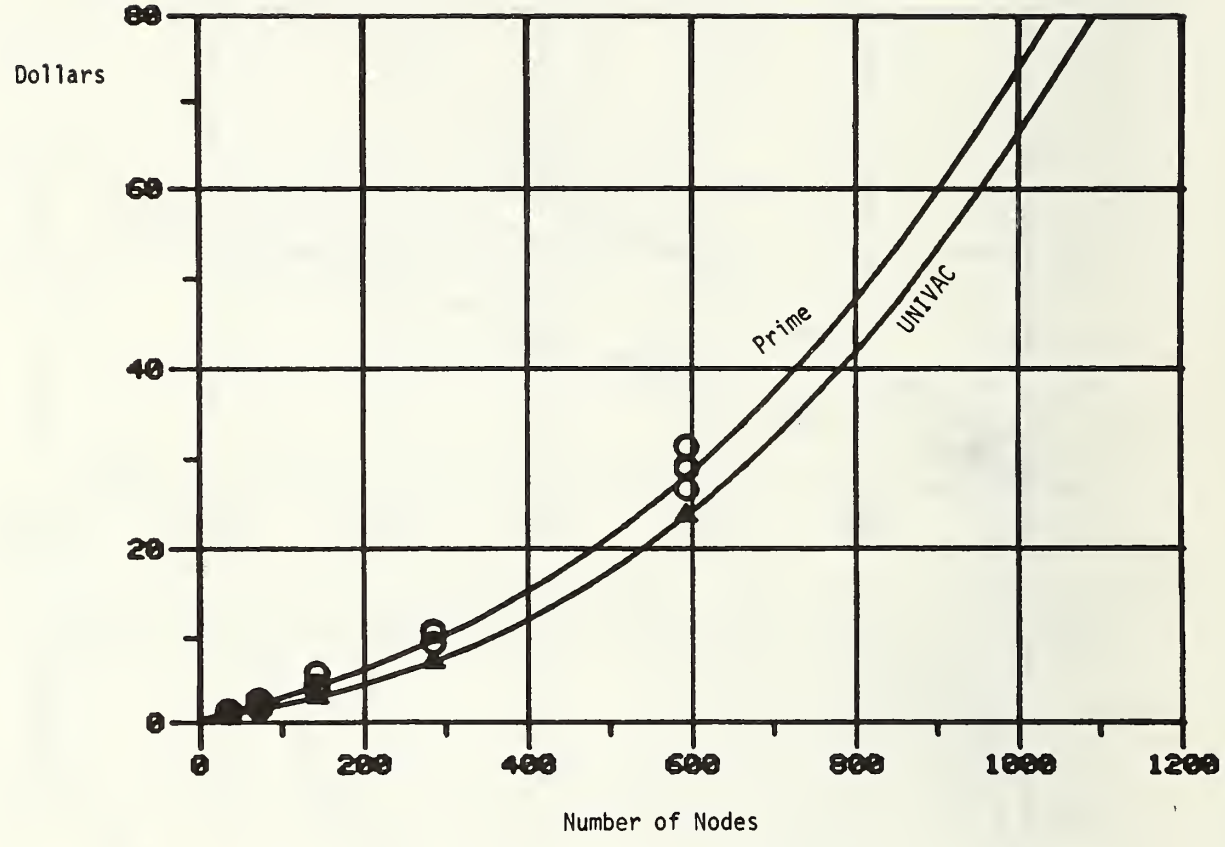


Figure 5. Comparison of total cost for the plate problem using the SAP IV program

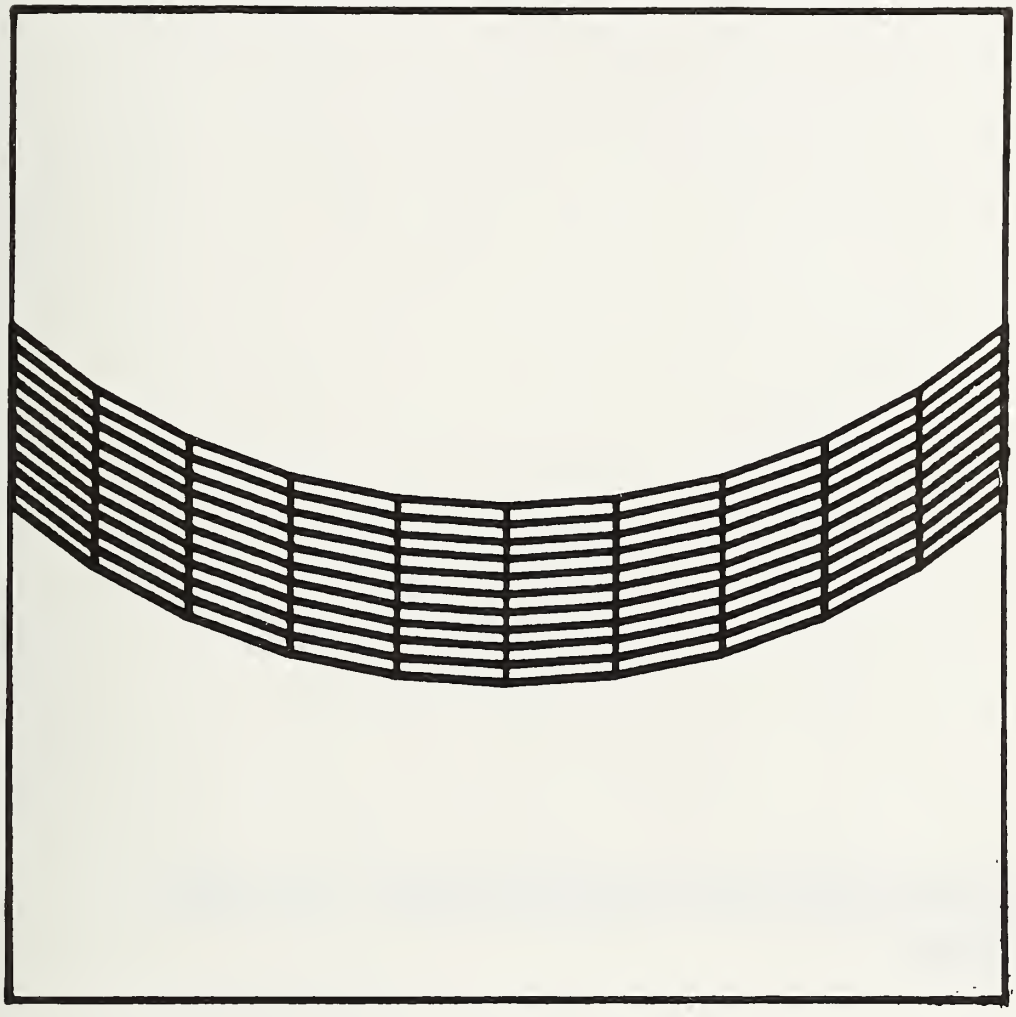


Figure 6. A typical cylinder model

Special Technologies and Applications
Track B

CPEUG81

Focusing on Secondary Storage

SESSION OVERVIEW

FOCUSING ON SECONDARY STORAGE

H. Pat Artis

Morino Associates, Inc.
Vienna, VA 22180

During the past decade, secondary storage has been the fastest growing resource for most installations. These devices now dominate the physical planning process and they represent an ever more significant portion of hardware budgets. The papers in this session address three secondary storage management topics. They are:

- o user experiences with an automated space management system. These systems attempt to reclaim space occupied by datasets that have been unreferenced for long periods of time.
- o guidelines for designing IBM I/O configurations to insure system performance.
- o an analysis of the architecture and performance characteristics of the CDC 844-41 disk subsystem.

Each of these papers provide a valuable perspective and the session should be informative for those who attend.

EXPERIENCES WITH DASD, TAPE, AND MSS SPACE MANAGEMENT USING DMS

Patrick A. Drayton

Southwestern Bell Telephone Company
St. Loius, MO 63101

This paper was written to share our experiences in the space management of DASD, Mass Storage and tape datasets in order to reduce DASD cost and improve the performance of the I/O subsystem. Included is a description of the situation which precipitated our turning to space management as a solution. Also included are the standards and procedures we introduced and the software products we utilized and wrote to service our needs. Finally the results that were achieved and some unforeseen problems that arose are documented.

Key words: data set, migration, DASD, MSS, space management, performance, disk management system.

1. System Configuration

The system under discussion was an IBM 3033AP with 16MK running under MVS-SE2. It was configured with 73 3350-type DASD and 4 STC 8360 DASD. In addition, it utilized 12 tape drives, one 2305 drum and one 100 GIGABYTE Mass Storage System (MSS) with four readers. The drum and 8360s, were dedicated to local page data sets.

The system served as the corporate test system and was used by 800 programmers from primarily 7 A.M. - 5 P.M. The average daily workload consisted of 1700 batch jobs in 25 initiators, 65,000 TSO transactions processed by an average of 70 on-line users, plus a small test IMS system with two message processing regions.

2. Problem Definition

The problems came to light in October 1980 when the 3033 was upgraded to an attached processer (AP) but the user community did not experience any measureable service improvements. The CPU could not be driven above 65% busy and batch thruput was below our service objective of 80% jobs turned around in under two hours. TSO response time on average was acceptable (2.2 seconds) but intermitant responses of 2-5 minutes were not unusual.

Initial investigation identified the problem as the MSS. Its four readers were busy 90-100% of the time with an average of six data sets awaiting service, this caused data sets to require over a minute from initial request to accessability on the staging packs. This resulted in jobs being swapped out for an average of 4.3 minutes of a total average elapsed time of 11.5 minutes. In other words 37% of the time a job was awaiting data set servicing. On comparing long wait swap times on MSS and non-MSS systems it was estimated that 80% of the long wait time was directly attributable to the MSS related

functions. Further investigation uncovered the fact that the operating systems method of new data set allocation caused severe degradation to MSS performance. During MSS space allocation no additional new or old data sets could be allocated even if they were already staged until after the initial new space allocation request was satisfied. MSS volumes are considered mountable devices and MVS allocation will stop all like-device allocations until the current allocation recovery is resolved. With these problems in mind we felt the improvement of MSS service was the number one priority item to improving overall service to the user community.

In the process of developing solutions to improving MSS service, additional problems were uncovered. One of these was extensive use of the private pack concept by application programming groups. That is, each application was given a number of dedicated DASD volumes to use as they saw fit with no guidance or policing. This was an outgrowth of capacity planning which required applications to justify their DASD requirements. These requirements were translated into DASD volumes and distributed as dedicated packs. The problem we found with this concept was the application programmers were not space managers and the packs contained old data, over-allocated data sets, and were badly fragmented.

Also the shared DASD pool was badly fragmented, it contained many out-of-date data sets and the current data sets were badly over-allocated and poorly blocked. This had resulted in uncontrolled growth of DASD requirements. When the user community could not allocate adequate real DASD they turned to MSS for their data space needs. This further aggravated the MSS busy condition by placing many highly active partitioned, TSO data sets and indexed files on MSS.

Two additional problems soon became evident. There were no data set naming standards in place. Data sets were named at the whim of the user so that the ownership could not be ascertained. In addition no data set placement guidelines had been issued. The problem this presented to MSS was previously described. In addition this caused system and user data sets to be spread throughout the real DASD pool. Data sets were placed by the user wherever space was available without regard for DASD contention or data set performance.

3. Corrective Action

Now that the problems had been identified we set about to relieve them. Our solutions are summarized as follows:

1. Disk Pack Reconfiguration Plan - this plan grouped functionally-related data sets together, removed the private pack concept, and configured packs based on performance.

2. Data Set Naming Standard - in order that ownership could be established and to allow data migration the data set name had to be in a standard form and this standard had to be automatically enforceable.

3. Data Set Placement and Migration Plan - this plan established where data sets should be placed based on their data set characteristics and usage activity. In addition, a criteria for moving data sets after their creation if their usage activity changed was required. Finally an automated method of accomplishing the above was essential to minimize manpower requirements.

3.1. Disk Pack Reconfiguration Plan

As described earlier there existed a need to configure data sets by usage to allow better control and performance of the DASD subsystem. One of the first criteria implemented was the termination of the private pack concept. This was met with resistance from the user community but was overcome with upper management backing plus the commitment that future user data space requirements would be met.

All DASD was then divided by function as follows:

1. Scratch/work volumes - these nine (9) volumes contained temporary data sets which would be deleted at job termination.

2. System data sets - all data sets containing system-required modules plus system-wide data sets such as SMF, JES SPOOL, JES checkpoint, and page/swap data sets were located on these 13 volumes. Page and swap data sets were on dedicated volumes.

3. Application libraries - each application's object, load and source libraries were contained on these six (6) volumes. Because of their low usage levels source libraries were used to fill the space remaining on local page packs. We found this was an excellent use of this space and did not cause any paging service degradation.

4. TSO data sets - these five volumes contained TSO edit type data sets. In order to insure space was available for TSO edit usage all data sets not used in 14 days were deleted and user data sets older than 28 days were copied to user DASD (see item g).

5. VSAM data sets - these three (3) volumes contained volume-size VSAM data spaces. This eliminated hundreds of user VSAM data spaces spread throughout DASD. This also allowed for more accurate space management of VSAM data sets. Our initial survey found 45% of the VSAM data sets were either empty or unusable.

6. Test data bases - frequently we had requirements for 1-4 volumes of DASD to perform volume tests or conversion work for an application. In the past this precipited a mad scramble attempting to secure unused volumes from other systems. This was replaced by a pool of four 3350s; which were scheduled on an as needed temporary basis. Requests for a total of more than four volumes from multiple users caused volume sharing. For example one application would have the volume for twelve hours a day. At the end of their test period the pack would be dumped and then restored with the next user's data.

7. User data sets - the vast majority of DASD was contained in this pool. All user test data sets, libraries, and data bases were located on these 39 volumes. These were the only volumes to participate in data set migration.

8. New MSS allocation staging - these three (3) volumes contained all newly created permanent data sets. During nightly maintenance the data sets on these packs were migrated to either real DASD (user data set volumes) or MSS. This was done to avoid the degradation new data set allocation caused on the MSS and to allow efficient placement of user data sets on real DASD.

This new DASD configuration greatly enhanced our ability to control space requirements and I/O performance.

3.2 Data Set Naming Standard

The naming standard was beneficial in locating the owner of a data set when problems arose. It also made possible the accounting for space usage back to the user department. This in turn allowed for better capacity planning of future DASD and MSS requirements.

However, the naming standard was essential for continuation of the MSS group concept after data set migration was started. MSS performance is enhanced by the use of the group concept. This concept causes the volumes of MSS to be logically connected and space requests to be directed to a group name and be satisfied by any volume in that group (similar to a generic name). We chose to assign groups by application. Therefore, any migration to MSS must go to a particular group. The data set name must, therefore, correctly identify the application owner.

The standard we implemented was as follows:

Each data set contained two index levels "DD.dddt" where DD is the department level (we have three) ddd is the division level within that department, t is the district within that division.

This standard matched our current TSO user ID scheme of DDdddtu with u representing the individual user. All data sets created over TSO by a user are automatically supplied the two standard indices. In addition, for performance we installed three user catalogs based on the department level qualifiers. As an aside, many of our users added a third level index of their initials to allow them to interogate the catalog for their unique three levels to obtain a listing of their data sets. Enforcement of our standard was accomplished through the use of DMS (see below).

3.3 Software Products

There were two software products used to take the data placement/migration plan from paper to reality. It will be noted in each portion of the plan where a particular tool was utilized. Following is a brief synopsis of the software used.

1. DMS-(Disk Management System) by Application Development Systems, Inc. This was our primary migration tool and our standard enforcer. This product accomplished the following:

   a. Movement of data to and from MSS and DASD.
   b. Archival to tape of old data sets.
   c. Deleting of old data sets.
   d. Deleting of non-standard named data sets.
   e. Releasing of over-allocated space in sequential and partitioned data sets.
   f. Deleting of non-catalogued data sets.

2. SAS - Statistical Analysis System by SAS Institute Inc. We made use of this wide-spread data tool as follows.

   a. Process SMF data to report on data set activity.
   b. Process RMF data to report on DASD pack and string contention.
   c. Process MSS trace data to report on MSS performance.
   d. Produced control card data to direct DMS processing.

3. Local Programs - local programs were required to build control cards for DMS to handle the migration from DASD to MSS.

3.4 Data Set Placement Plan

The purpose of this plan was three-fold:

1. Reduce the load on the MSS readers by migrating busy data sets from MSS to DASD.
2. Reduce the need for increased real DASD by migrating unused data sets to MSS.
3. Avoid the degradation of new allocation on MSS by delaying the placement of new data sets until nightly maintenance.

The placement standards were established as follows:

1. DASD - frequently accessed (more than 15 times/week) user data sets; system data sets, ISAM data sets, application group object, load, and source libraries; large (over 50 MB) IMS, VSAM data bases.
2. MSS - Intermediate access (1-14 times per week), intermediate sized (0-50MB) data sets. The MSS cartridge holds 50MB of data).
3. Tape - not frequently accessed (less than once/week), large sequential data sets (larger than 200 MB), system log files, archived data sets.

The DASD placement of system and application libraries was handled in the reconfiguration plan. In addition at this time each application's libraries were given a 20% space increase for growth and placed on DASD

so that no further expansion of these libraries was possible without the space manager's knowledge. The application programming staffs were then made accountable for their library contents and insuring that they remain within the established space limitations. Large data sets and IMS data base were placed as specified by the space manager. All new large space requests were handled by the space manager. VSAM data sets were placed in the common VSAM data spaces.

MSS placement was handled by the introduction of a new-MSS DASD pool and the elimination of direct new space requests to MSS by all but system maintenance jobs. This was accomplished through the use of JES and TSO exits to restrict the use of the MSS unit parameter. The users placed all new non-TSO, non-tape type data sets on the new-MSS DASD volumes and during nightly maintenance the data sets were either copied to real DASD or MSS based on the following usage criteria: A data set used four times in two days was copied to DASD. Otherwise it was moved to MSS.

Tape data set placement was basically the catch-all for data sets that were too large for MSS or DASD.

3.5 Migration Procedure

The final phase of the space management solution was the development of a migration procedure which would reduce the load on MSS and remove uneeded DASD data sets without inconveniencing the user community. Figure 1 is a summation of the final migration procedure parameters.

MIGRATION PROCEDURE

DASD - MSS
NOT ACCESSED IN FIVE DAYS
DMS AND LOCAL PROGRAM USED

MSS - DASD
ACCESSED 15 TIMES IN 7 DAYS
SAS AND DMS PROGRAM USED

DASD - TAPE
DATA BASES NOT ACCESSED IN TWO MONTHS
MSS - TAPE
NOT ACCESSED IN SIX MONTHS

TAPE - DELETED
NOT ACCESSED IN ONE YEAR

FIGURE 1

3.5.1 DASD to MSS

Smaller (less than 50MB) data sets were migrated to MSS if they had not been accessed in five days. This resulted in an average of twenty data sets being migrated daily. This figure would have been larger but we were not able to keep up with the growing demand of MSS space. For economy reasons MSS cartridges were not ordered until they were needed. Therefore our MSS had only 42% of the allowable cartridges installed but the cartridges installed were 67% allocated. DMS was used to identify those data sets not accessed in five days and a local program was written to read the MSS control table and to assign MSS group identification based on data set name. It also supplied control cards to DMS so that it would perform the copy.

3.5.2 MSS to DASD

All MSS data sets accessed 15 times in the past seven days were migrated to DASD. This was done so that the delay from a data set being heavily accessed and its migration to a faster device would be minimal but also avoid the migration of every MSS data set that was accessed. This resulted in an average of ten data sets being migrated daily. SAS was utilized to read SMF data to identify heavily accessed MSS data sets and to build control cards for DMS to perform the copy.

3.5.3 DASD to Tape

Larger data sets (over 50MB) not used for two months were archived to tape. If required, the user had to request a restore via a TSO command and the data sets was returned during nightly maintenance. This was a slight inconvenience but one day turnaround for data inactive for over two months was not felt to be excessive.

3.5.4 MSS to Tape

MSS data sets not accessed in six months were archived to tape. This was caused because of the pressure of cartridge growth. Again, the user needed only to request a restore on TSO to retrieve his data set.

3.5.5 Tape to Delete

Tape data sets, which had their retention periods exceeded were deleted. In addition, DMS archived data sets which were over one year old were deleted. For MSS data sets that were archived after six months, this involved a six month retention on tape. For DASD data sets archived after two months, this involved a ten 10 month retention.

4. Results

The total effort from the perception of the problem to the attainment of improved results was six months. The total manpower required was one person full time and two people for 25% of their time. Figure 2 summarizes the outcome of this effort.

RESULTS

	Before	After	Change
MSS READER BUSY	95%	57%	-38%
MSS DATA SET STAGE TIME	62SEC	27SEC	-35 sec
MSS DATA SET QUEUE	6.0	1.1	-4.9
JOB LONG WAIT TIME	4.3MIN	2.1MIN	-2.2 min
JOB ELAPSED TIME	11.5MIN	7.9MIN	-3.6 min
JOB INPUT Q TIME	65MIN	20 MIN	-45 min
TSO RESP TIME (TRIVIAL)	2.2SEC	1.5SEC	-.7 sec
TSO TRANS-ACTIONS (8-5)	55K	67K	+12K
JOBS PROCESSED (8-5)	1100	1700	+600
CPU BUSY(8-5)	65%	85%	+20%
DASD ADDITION (6 MON)	8	2	-6

FIGURE 2

As oulined previously, the MSS device was a bottleneck to good system performance. The migration/placement efforts reduced the MSS reader busy from 95% to 57% with the IBM guideline value issued as 70%. More importantly, the service provided by MSS improved as measured by average data set stage time reduced from 62 to 27 seconds and average number of MSS data sets awaiting service reduced from 6.0 to 1.1. This in turn was reflected in improved batch performance by reducing long wait swapped out time from 4.3 to 2.1 minutes and average job elapsed time from 11.5 to 7.9 minutes. This also effects the batch service queues by reducing average job input queue time from 65 to 20 minutes. The TSO system also benefited as evidence the reduction from 2.2 to 1.5 seconds in TSO trivial response time while the

workload executed by TSO increased from 55,000 to 67,000 transaction per day. This increased workload was also evidenced in the increase in CPU busy from 65 to 85%.

Two notes on those results -

1. All of this was accomplished with an increase of only two DASD volumes.
2. The greatest performance improvements were realized by eliminating the allocation of new MSS data sets directly to MSS (i.e. reader busy was reduced from 80% to 57% after this was implemented) and the dedication of 8360's as local page data sets (this increased the CPU busy percentage from a level of 75% to a level of 85%).

We are satisfied that the results experienced justified the effort expended.

4.1 Continuous Monitoring

In order to insure a good system performance, continuous monitoring of the DASD/MSS environment was done. This included weekly reviews of:

1. DASD channel and string usage using RMF data.
2. DASD pack usage and queueing using RMF data.
3. Allocated and available DASD space using DMS reports.
4. MSS available space using IDCAMS utility reports.
5. MSS reader busy and data set staging performance using SAS analysis of MSS trace files.

Any time the data indicated a problem in a particular area corrective action was taken immediately. This prevented small performance problems from becoming major system bottlenecks.

4.2 Experiences

Not always was it obvious that the end would justify the means. There were times where it was doubtful any progress could be made. I felt it would be informative to relate some of our more interesting experiences.

1. Data set naming standards - the users were given six weeks notice that only standard names would be retained - all non-standard data sets would be deleted. The user community said there was no way they could handle that change in under one year. Management supported our decision and we cut over as planned. Suprisingly, there were fewer than ten requests to restore data sets of the more than 4000 that were scratched. That indicated to us the larger number of unused data sets on DASD and MSS.

2. Private packs - next to the naming standards the elimination of private packs was the least liked of our changes. But we found that if the needed space was provided when the user required it that was all that the user desired. They didn't relly want or need their own volume.

3. Reference date updates - since DMS used a data set's last reference date for migration purposes it was imperative that this be accurate. We found that soon after we implemented our plan users developed programs that opened all their data sets on a weekly basis in order that they remain on DASD. We have caught a few users doing this and stopped them. We have also coded programs that process SMF data to identify this activity and report it to upper management.

4. PDS directory problems - with our release of MVS (SE2) there occurred a problem. DMS updates the VTOC each time a data set is opened;the reference time, date, and jobname. If two users are updating the same partitioned data set, a problem can occur of having overlapped VTOC updates which are not replaced in the correct sequence. After losing track of a few partitioned data sets, we turned DMS tracking of system-wide partitioned data sets off.

5. MSS maintenance procedures - due to the way MSS picks and stages each data set, any maintenance which operates at the data set level is very time consuming. DMS operates at this level, and it takes about 12 hours to do a DMS backup of 30 MSS volumes and three (3) hours to scan 100 MSS volumes for invalid data set names. For this we chose the incremental backup featyre if DMS whereby only changed data sets are backed up. This dramatically reduced our maintenance time.

6. Large record length on MSS - a data set with a record length larger than track size (13030 bytes) could not be placed on MSS. This occurred because the DASD (3350) maximum is (19069 bytes). We had no solution to this problem and were forced to move these data sets to tape after two months of non-use.

7. Non-catalogued data sets - all of our migrations from MSS to DASD and vice versa assume that the user can find the location of his data set by means of its catalog entry. For that reason all volume references had to be removed and all non-catalogued data sets were deleted.

8. Empty data sets - when the initial migration concept was announced many users sought to guarantee DASD space by allocating empty data sets just in case they would need it. DMS released over-allocated space but as of this writing could not free totally empty data sets. This resulted in a manual process of reviewing listings and deleting empty data sets.

9. APF libraries - data sets in the Authorized Program Facility have their volume serial number listed. Movement of these data sets caused system problems. We had to exclude these from our migration.

10. Release over-allocated. Through the use of DMS we released over-allocated space in partitioned and sequential data sets. We averaged two volumes of space released during every weekly run. This process required less than fifteen minutes per week.

5. Conclusion.

In summary there are three areas which were vital for the success of this project. They were communication to the user community, management commitment; and credibility.

Throughout our effort we continually communicated with the user via letters to their management and bulletins to the programmers as to what we were doing, when the change was effective, why things were changing, and the effect of the change. To this end we made extensive use of an already established user group. This user group contained one representive from each application and served as our liaison to the users.

There were numerous times where the user complaints of changes required could have turned management off to our project. But fortunately they stuck with us and weathered the storm. With their backing the changes were made, the standards were followed, and our project was a success.

The fact that we showed the user that we could deliver service was very important. After placing all those restrictions on them if we had not been able to accomodate legitimate space requests all of our efforts would have been for naught. I feel this was the most important concept of making a data set placement and migration plan work-credibility with the user community.

I would like to acknowledge the efforts of Greg Marshall, in trouble shooting DMS problems and Jeff Wides, for sharing his MSS expertise and performance data.

DASD CAPACITY PLANNING
3350 DASD

Keith E. Silliman

IBM Corporation
Federal Systems Division
18100 Frederick Pike
Gaithersburg, MD 20879

A pragmatic approach to Direct Access Storage Device (DASD) I/O capacity is viewed as the ability to do I/O in terms of the number of I/O operations and bytes of data transferred. Maximum and "average" I/O capacity algorithms are developed for 3350 DASD. Data transfer and density of reference capacity values are derived. Examples of capacity value usage are given.

Key Words: DASD I/O Capacity, Density of Data Reference, I/O Capacity Algorithms, 3350 DASD.

INTRODUCTION

There is an increased need to understand the capacity of Direct Access Storage Devices (DASD) to do I/O, in terms of the number of I/O operations or data bytes transferred per unit of time. DASD technology has made tremendous advances in increasing data density. The resulting reduction in the cost per megabyte has made possible increased amounts of "online" data. Those increased amounts of online data, the fixed size of existing computer facilities, and the increasing floor space costs associated with new facilities have necessitated even higher density DASD.

Sizing of DASD subsystem based on the number of megabytes of data to be stored, floor space, and available funds frequently produce configurations with the fewest number of the highest density DASD available. As a result, we believe there is an increased awareness of the performance impact of the I/O subsystem [1,2,3,4,5,6] and considerable emphasis is being placed on capacity planning [7,8,9].

DASD capacity to do I/O is a critical consideration in the design and tuning of high performance computer systems, especially online, data retrieval or Data Base Management Systems (DBMS). Increasing amounts of data are stored, accessed and retrieved from DASD. Responsive user interactions require an adequate DASD subsystem. The design of the DASD subsystem, inturn, depends on definitive system requirements and knowledge of device characteristics and capacities. This paper explores the capacity of the 3350-type DASD to do I/O in an IBM 3/370 MVS environment.

3350 DASD

The IBM 3350 [10] is a large capacity, fast access, high data rate, Direct Access Storage Device unit. A unit consists of two drives, each with a head/disk assembly (HDA) with 16 recording surfaces. Each drive has a data storage capacity of 317.5 million bytes, allocated in 16,650 tracks (19,069 bytes per track) and grouped into 555 cylinders (30 tracks per cylinder). The disks rotate at 3,600 revolutions per minute, sixty revolutions per second, or once every 16.7 milliseconds. A minimum of 10 milliseconds is required to move the access mechanism (read/write head). The maximum head movement time (seek) is 50 milliseconds. The data transfer rate is 1.198 million bytes per second.

The 3350 is a Rotation Position Sensing (RPS) device [11]. It logically disconnects from the channel following the set sector command and waits for the specified sector to come under the read/write head. That waiting time is the rotational delay time. When the specified sector comes under the read/write head, the device attempts to logically reconnect with the channel. Following a successful reconnect, data transfer commences. At the end of data transfer, a simultaneous channel end/device end I/O interrupt is presented to the CPU, completing the DASD portion of the I/O operation.

If at I/O initiation time the read/write head is not positioned to correct cylinder, a seek occurs to move it. During the seek, the I/O pathway is available to service other requests. At seek completion, the channel is notified and the set sector command is issued.

If the device cannot logically reconnect because a pathway component is busy (i.e., head of string, control unit, or channel), a missed reconnect occurs and a full rotation (16.7 milliseconds) must complete before another logical reconnect is attempted. All the while, the device is busy "processing" the I/O operation.

3350 CAPACITY MEASUREMENTS

Measurements of I/O activity provide insight to DASD I/O capacity. A plot of 3350 device busy and I/O operations per hour is presented in Figure 1. The data plotted are one-hour samples for individual 3350 devices that exhibited device utilization greater than 10 percent. (The 10 percent threshold was an arbitrary choice to reduce the total number of samples plotted.) They are from a one-week period on two IBM S/370 158 attached processor (mixed online and batch workload) systems and an IBM 3033 (batch) system. The maximum rate for a one-hour sample was approximately 127,000 I/O operations, or 35 per second, and a 72 percent device utilization.

The plot in Figure 1 indicates an average device service time of approximately 20 milliseconds (derived by multiplying the device utilization by the number of seconds per hour and dividing by the number of I/Os per hour). Samples in the lower right have shorter device services times, indicating small effective data block sizes and/or minimal seeks and missed reconnects. Samples in the upper left have longer device service times, indicating larger effective data block sizes and/or increased number or duration of seeks and missed reconnects.

A projection of device utilization to 100 percent with a device service time of 20 milliseconds indicates a device capacity of 180,000 I/O operations per hour or 50 per second. If the device service time was 16.7 milliseconds and 100 percent busy, the capacity would be 216,000 I/Os per hour or 60 per second (one per revolution).

However, the elongated queue delay effect as a function of resource utilization (response time = service time/(1-resource utilization) will cause I/O response time degradation with higher device utilizations.

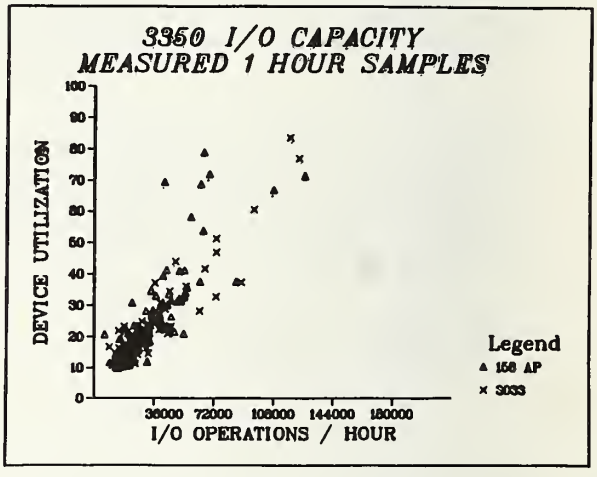


FIGURE 1 3350 I/O CAPACITY MEASURED 1 HOUR SAMPLES

A measure of queue delay is plotted with device busy for the same samples and is presented in Figure 2. That plot indicates an increase of queue delay with increased device utilization but with some significant deviations. Samples of relatively high device utilization and low queue delay occur, implying synchonized arrivals, probably a single user on the device.

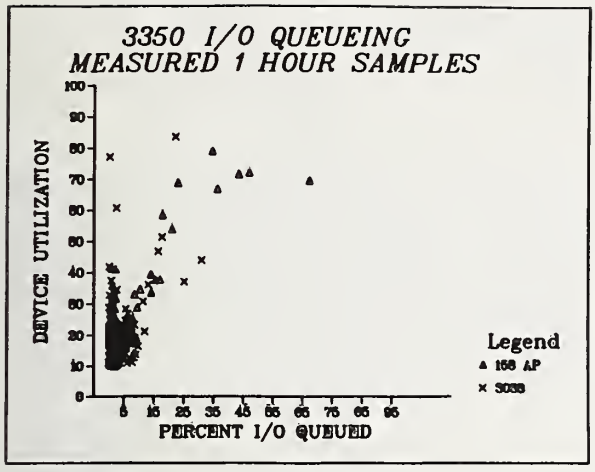


FIGURE 2 3350 I/O QUEUEING MEASURED 1 HOUR SAMPLES.

General guidelines, or rules-of-thumb, for performance tracking and comparison have been developed. Batch system environment values are 35 percent device busy and an average queue length of 0.1 [2,3]. Data base system values are 30 percent device busy and average queue length of 0.04 [4]. Measured I/O activity rates for 30 to 35 percent device utilization are 36,000 to 72,000 per hour or 10 to 20 per second (see Figure 1). Up to 15 percent of the I/O operations are queued for 30 to 35 percent device utilization (see Figure 2).

DASD I/O

A DASD I/O operation is the sum of a combined effort of the CPU, channel, control unit, head of string, and device. The sequence of events is as follows: A software task requests a DASD I/O operation by executing the Execute Channel Program (EXCP) code which includes a Start I/O (SIO) instruction. With the successful execution of the SIO (condition code = 0), the I/O subsystem portion of the operation is initiated. A seek may occur to position the read/write head to the specified cylinder. The set sector command is issued by the channel. The device revolves to position the requested data area under the read/write head. The device logically reconnects with the channel and data transfers. The simultaneous channel end/device end I/O interrupt is processed by the I/O Supervisor (IOS) routines in the CPU. Finally, the task that requested the I/O operation is informed that the data transfer has completed. That task then processes the data read or prepares additional data to be written and may repeat the process. A schematic diagram of a complete DASD I/O operation is presented in Figure 3.

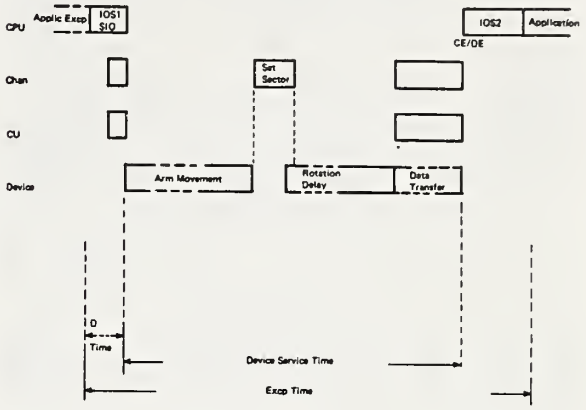


Figure 3. Schematic Diagram of DASD I/O Operation

Device service time is the elapsed time from successful SIO to data transfer completion [3,4]. The EXCP, or I/O time, is the device service time plus the combined CPU processing of the EXCP, any queue delay incurred initiating the I/O, and the I/O interrupt processing which follows data transfer.

The DASD I/O process may be interrupted or delayed at any of several steps. Higher priority tasks may preempt the CPU from the task attempting to execute the EXCP. A busy component of the logical channel (channel or device) can cause the request to be queued for later processing when the component is available [1]. An unsuccessful SIO (condition code = 1) will occur if the DASD control unit or head of string is busy even though the channel and device are not busy [3]. Tha condition occurs in a shared DASD or a channel/string switched environment. Seek frequency and duration are a function of data set placement, access methods used, and concurrency of device use. The duration of rotational delay is primarily a function of missed reconnects which occur because of I/O pathway component busy.

Our experience indicates that DASD I/O times (from EXCP to EXCP) should average 100 milliseconds or less on S/370 Model 158 or larger systems. Five general areas contrib-

ute to that elapsed time: (1) CPU time required to execute the EXCP and process the I/O interrupt, (2) logical channel queue time waiting to successfully start the I/O, (3) seek time, (4) rotational delay time, and (5) data transfer.

The time required by the CPU to process a DASD I/O operation depends on the CPU speed (MIPS/MOPS) and the number of instructions executed. Measurements of MVS environments indicate 10,000 to 50,000 instructions per I/O operation [5] for supervisory and application functions combined. The number of supervisory instructions is fixed for a given access method. "In MVS 3.7 . . . the path length for an EXCP is approximately 3800 instructions" [12]. The number of application instructions depends on the data block size and the intensity of the scan of the data read or the build process for data written. The number of instructions divided by the CPU instruction processing rate provides the CPU time.

The logical channel queue time is the delay time spent waiting for the required I/O pathway components to become available to satisfy the request and is a function of the I/O subsystem utilization [6]. The magnitude of the delay will vary up to a multiple of the device service time when multiple requests are made to an individual drive.

The seek time delay will occur whenever the read/write head is not positioned at the correct cylinder. Seek times for an IBM 3350 are 10 to 50 milliseconds. Random access data will generally have one seek per I/O operation, whereas sequentially accessed data will generally have multiple I/Os per seek. On a contended DASD, even sequentially accessed data may have one seek per I/O.

Rotational delay time will occur for essentially every DASD I/O. The average rotational delay (not including missed reconnects) is 8.3 milliseconds (one-half of a revolution). Missed reconnects occur as a function of the I/O pathway component utilization [6]. Each missed reconnect adds 16.7 milliseconds to device service time and corresponding increases in I/O response time. Consecutive missed reconnects can, and do, happen.

Data transfer time depends on the data block size; the number of bytes transferred at the rate of 1.198 million bytes per second (roughly 1,000 bytes per millisecond). For direct access I/O using Basic Direct Access Method (BDAM), the effective data block size is the actual block size. Using search previous I/O, with either Basic Sequential Access Method (BSAM) or Queued Sequential Access Method (QSAM), may result in an "effective" data block size twice the actual block size [3]. Use of multiple buffers may also cause the effective data block size to be larger than the actual.

Three of the I/O component times (CPU time, seek time, and data transfer time) are fixed as a function of the hardware and software. Logical channel queue time and rotational delay time, including missed reconnects, are not bounded, especially in a shared DASD environment.

DASD CAPACITY DISCUSSION

The preceding discussion of 3350 DASD, DASD I/O characteristics, and the data presented in Figure 1 indicate that, in general, a maximum of one I/O operation can occur per device revolution, or 60 per second. Because of contention at the CPU and I/O pathway component level (seeks, missed reconnects, and variable data transfer times), the "average" capacity is significantly less then one transfer per revolution.

Consecutive data blocks can be read or written (if the CPU is fast enough to execute the required I/Os and application instructions to process the previous I/O interrupt, process the data, and initiate the next I/O operation) in consecutive device revolutions, assuming sequential BDAM reads or writes on one cylinder (no seeks) of a dedicated device, string, control unit, channel, and CPU. However, after the last data block on a track is transferred, the next revolution will occur without a data transfer. Following the transfer of the last data block on the track N, the CPU will be busy processing the I/O interrupt while the first data block on track N+1 is moving under the read/write head.

An analytical representation of that relationship for different data block sizes is presented in Figure 4. To read or write 10 consecutive one-tenth track size data blocks requires 11 revolutions. An average of five revolutions will be required to read or write four one-quarter track data blocks. Two consecutive half track blocks require an average of three revolutions, and an average of three revolutions, and an average of two revolutions will be required for each full track block.

DASD I/O CAPACITY ALGORITHM

Based on the assumption of the effective maximum of one I/O operation per device revolution and a missed revolution between the consecutive accesses of the last data

block on a track and the first on the next track, a Maximum DASD I/O Capacity Algorithm was developed. That algorithm is as follows:

This algorithm indicates the maximum capacity of a 3350-type DASD to be 216,000 I/Os per hour (60 I/Os per second) for very small data block sizes. Capacity decreases steadily with increased data block size to 108,000 I/Os per hour (30 I/Os per second) for half track blocks (9,300 bytes) and remains at that level to full track blocks 19,000 bytes). Data block sizes in excess of full track cause a further decrease in I/O capacity.

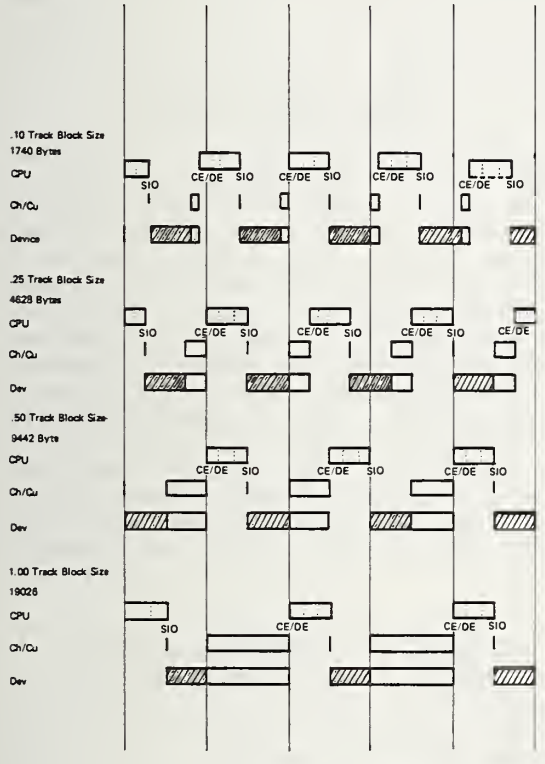


Figure 4 Accessing Consecutive Data Blocks

AVERAGE DASD I/O CAPACITY

The "average" capacity should be of primary interest to the computer systems analyst not the maximum capacity. Where average connotes practical or realistic values. The maximum rate projections assumed dedicated CPU and I/O subsystem components. DASD revolutions will be missed (i.e., go unused) in a contended environment because; (1) I/O operations cannot be initiated, (2) head movements (seeks), (3) missed reconnects, (4) and application tasks are not dispatched in a timely fashion. A minimum seek (one cylinder) requires 10 milliseconds, or more than half a revolution, a maximum seek requires 50 milliseconds, or three revolutions. A single missed reconnect causes a 16.7 millisecond (one revolution) delay.

"Average" DASD I/O capacity may be one-third (or less) of the mazimum capacity. An algorithm for Average DASD I/O Capacity, using the one-third value, is as follows:

$$\text{Average DASD I/O Capacity} = \frac{\text{Maximum DASD I/O Capacity}}{3.3}$$

This algorithm indicates the average capacity of a 3350 DASD to be approximately 65,000 I/O's per hour (18 I/Os per second) for very small data block sizes. With increased data block size, it decreases to 32,000 I/Os per hour (9 I/Os per second) for half track (9,300 bytes) data blocks and remains at that level to full track (19,000 bytes) block size. Figure 5 presents the algorithm results for maximum and average DASD I/O capacity for data block sizes from 1,000 to 19,000 bytes per data block.

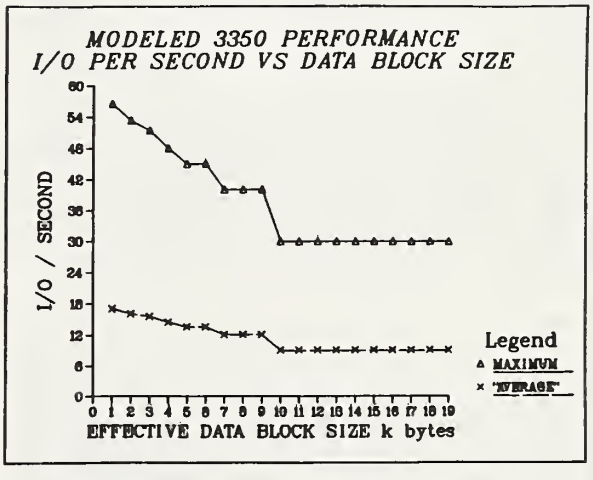


FIGURE 5 MODELED 3350 PERFORMANCE I/O PER SECOND VS DATA BLOCK SIZE

The I/O capacity values obtained from the algorithm are consistent with the measurement data presented in Figure 1 and the "rule-of-thumb" value of 30 to 35 percent for I/O component busy. Twenty-seven percent (62 of 229) of the samples in Figure 1 are between 32,000 and 65,000 I/Os per hour. Only six percent (14 of 229) exceed 65,000 per hour.

Those values are from a population of samples of unknown data block sizes that did not include any samples with device utilization less than ten percent.

The 35 percent device busy rule of thumb implies similar capacity values. For device service times of 20 milliseconds (approximately the value derived from the samples in Figure 1), the device capacity at 35 percent busy would be 63,000 I/Os per hour or 17.5 per second (.35 * 3600 / .020 sec). For 35 millisecond device service times, the device capacity would be 36,000 I/Os per hour or 10 per second (.35 * 3600 / .035 sec). For a constant device utilization value (e.g., 35 percent), the device capacity decreases as device service time increases.

DATA TRANSFER CAPACITY

Multiplying the derived DASD I/O capacity values by the corresponding data block sizes produces capacity data rates in bytes transferred. The maximum and "average" data rate capacities are plotted as a function of data block size in Figure 6. The maximum data rate capacity indicated is approximately 570,000 bytes per second (30 full track blocks) which is half the 1,198 million bytes per second instantaneous data rate. The corresponding average data rate capacity is 170,000 bytes per second (9 full track blocks). Smaller data block sizes produce a lesser number of bytes per second even though the number of data blocks transferred is larger.

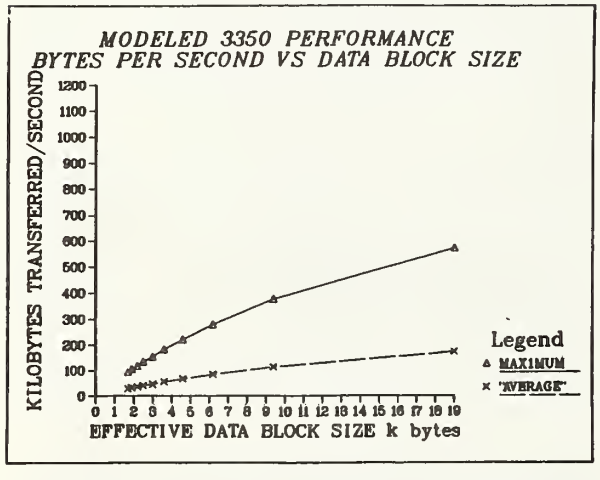


FIGURE 6 MODELED 3350 PERFORMANCE BYTES PER SECOND VS DATA BLOCK SIZE

DENSITY OF REFERENCE CAPACITY

Dividing the algorithm-derived I/O capacity values by the number of hundreds of megabytes of data stored on a 3350 (3.17) gives a measure of I/O capacity per 100 megabytes of data storage, or Density of Reference Capacity. The maximum and average 3350 density of reference capacities are plotted as a function of data block size in Figure 7. The maximum density of reference (per 100 megabytes) is indicated to be 18 I/Os per second for very small data block sizes, decreasing with increased data block size to 9 I/Os per second for half track and larger. The average density of reference is indicated to be 5 I/Os per second for very small data block sizes, decreasing with increased data block sizes to 3 I/Os per second at half track and larger data block sizes.

CAPACITY VALUE USAGE

DASD I/O capacity values can be utilized as guideline, or rule-of-thumb values. For example, while doing a data flow analysis of a planned response-oriented computer system, the implicit requirement to accomplish 18-20 I/Os per second (at 4 k bytes each) to a 3350 file becomes apparent. A comparison with the 3350 DASD capacity values indicates that 18-20 I/Os per second is within the capacity of the device but above the average for that data block size. Project management should be alerted to a possible performance problem. Cost/ performance trade studies should be initiated to assess the benefit/impact of increasing the data block size or distributing the data file (and accesses) across multiple devices.

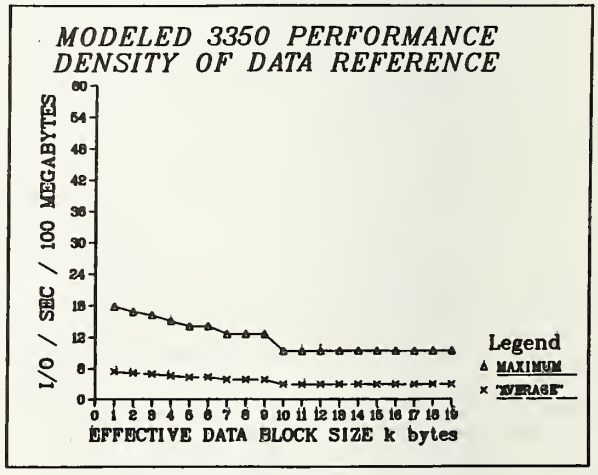


FIGURE 7 MODELED 3350 PERFORMANCE DENSITY OF DATA REFERENCE

The planning process for the migration of data from 3330-1 to 3350 DASD could use these values. I/O activity rates on the existing DASD are either known or measurable values. For 3330-1 DASD, the I/O rate measures are density of reference values per 100 megabytes because of 3330-1 is a 100 megabyte device. Comparison of those 3330-1 values with the derived 3350 values indicates whether the I/O capacity requirements of the data to be moved exceeds the capacity of the intended 3350 DASD. Average density of reference of the volumes to be migrated should not exceed the projected value for the receiving device. If the I/O rate to existing 3330-1 volumes exceeds the projected values for 3350s, the migration may result in increased I/O contention and degrade I/O response times.

Systems tuners should compare these capacity values with their system measurements. Significant deviations, either below or above these values, warrant further investigation. Systems not completing work or providing unsatisfactory response times should be expecially scrutinized. If sufficient workload exists, low activity rates (less than 5 I/Os per second) on the most active devices may indicate excessive head movement or I/O pathway contention problems. Values above the average may indicate too much data per volume (data base) or insufficient number of devices (paging and public work space). Systems with heavily accessed volumes (10 to 20, or greater, I/Os per second) and a CPU with a significant amounts of wait time (20 to 30 percent or greater) need to have the access frequency/ contention reduced by distributing the data accesses across more devices.

SUMMARY

We believe DASD capacity should mean more than the number of bytes of data stored on the device. Computer system analysts and managers need to become sensitive to DASD capacity to do I/O, the number of I/O operations, and the bytes of data transferred per unit time.

A discussion was presented indicating the combined, coordinated interaction of the CPU and I/O subsystem components required to accomplish DASD I/O. The speed of the CPU does not contribute to DASD I/O capacity or impairment thereof as long as the combined instruction execution time of I/Os and application code and any resultant CPU queue delay is less than the revolution period of the DASD (16.7 milliseconds for 3350s). CPUs with less capacity than a S/370 Model 158, application functions with a high level of data manipulation, or a high multiprogramming level may have 3350 DASD I/O capacities lower than indicated here.

A 3350 DASD I/O capacity algorithm that quantifies DASD I/O capacity, I/Os per second, and bytes transferred per second has been developed. Maximum and average capacity predictions were made. We believe this algorithm can be generalized to all DASD but have restricted our presentations and discussion to 3350-type DASD. Units of measure and predicted values are presented for DASD density of reference. They may be helpful in migration of data to higher density DASD.

The effective data block size and DASD track size are critical components of DASD I/O capacity. In general, the maximum number of I/O operations per second occur with the smallest block sizes where the CPU and data transfer times are minimized. For small data block sizes, the majority of the resource utilization burden is on the CPU. Increased data block size cause additional CPU costs to process the data as well as increased transfer times. However, CPU utilization decreases because I/O operations occur less frequently, whereas I/O pathway component utilization increases. The overall time required to process a given amount of data is reduced because of the dramatically increased number of bytes of data transferred per second.

The number of blocks of data per track determines the maximum I/O capacity of a DASD. Missing the next consecutive data block in a sequential I/O operation following the last block on the track becomes increasingly frequent with increased block size. For data block sizes greater than half track, essentially every data block is the last one on the track.

The I/O capacity measures of DASD I/O operations per second may be a more meaningful guideline than the 30 to 35 percent device utilization rule-of-thumb values. Average 3350 DASD I/O capacity is predicted to be between 9 and 18 I/O operations per second, depending on the effective data block size (for one-hour intervals). Unknown data block sizes can be approximated (channel service time multiplied by the device transfer rate).

References

1. K. Ziegler, Jr., "DASD Configuration and Sharing Considerations," Technical Bulletin GG22-9052, IBM Corporation (1978), Available through the local IBM Branch Office.

2. 'OS/VS2 MVS Performance Notebook,' GC28-0886, IBM Corporation (1979), Available through the local IBM Branch Office.

3. R. M. Schardt, "An MVS Tuning Approach," IBM Systems Journal 19, No. 1, 102-119 (1980).

4. S. Holt, "Information Management System - Virtual Storage - Multiple Virtual Storage (IMS/VS/MVS) Performance and Tuning Guide," Technical Bulletin G320-6004, IBM Corporation (1980), Available through the local IBM Branch Office.

5. J. B. Major, "Processor, I/O Path and DASD Configuration Capacity," IBM Systems Journal 20, No. 1, 63-85 (1981).

6. S. E. Frisenborg, "DASD Path and Device Contention Considerations," Technical Bulletin G22-9217 (1981) IBM Corporation, Available through the local IBM Branch Office.

7. L. Bronner, "Capacity Planning: An Introduction," Technical Bulletin GG22-9001, IBM Corporation (1977), Available through the local IBM Branch Office.

8. L. Bronner, "Capacity Planning: Implementation," Technical Bulletin GG22-9015, IBM Corporation (1979), Available through the local IBM Branch Office.

9. 'Capacity Planning Extended (CPX),' SB21-2392, IBM Corporation (1981), Available through the local IBM Branch Office.

10. 'Reference Manual for IBM 3350 Direct Access Storage,' GA26-1638, IBM Corporation (1977), Available through the local IBM Branch Office.

11. D. T. Brown, R. L. Eibsen, and C. A. Thorn, "Channel and Direct Access Device Architecture," IBM Systems Journal 11, No. 3, 186-199 (1972).

12. C. C. Burns, "Memory Can Increase Your Capacity," Technical Bulletin GG22-9053, IBM Corporation (1981), Available through the local IBM Branch Office.

AN ANALYSIS OF A CDC844-41 DISK SUBSYSTEM

J. William Atwood[1]
Keh-Chiang Yu

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

In contrast with IBM control programs, input/output supervisor software for CDC systems frequently issues transfer requests for entire program images (e.g., when loading or rolling out a program). These program image transfers have substantially larger service times than regular input/output accesses. A detailed simulation model has been used to show that, for CDC systems, significant performance advantages normally accrue when the access path to the rollout files is disjoint from the path(s) used for regular input/output accesses. The system modelled has a 12-spindle CDC 844-41 disk subsystem, which is shared between two CDC Cyber 170/750 central processors. Projections for a 20-spindle subsystem are also reported.

Keywords: CDC 844-41; computer system evaluation; disk subsystem configurations; input/output; modeling; rollin/rollout files; simulation.

1. Introduction

After an upgrade in central processor capacity at the Computation Center of the University of Texas at Austin (UTA), a careful measurement study was instituted to evaluate the performance of the new configuration. Event-trace data were recorded from the system, and a simulation model was validated. The model was used to project the performance of the system if the file allocation algorithms were altered in order to minimize the expected seek time for the disk subsystem. The observed data and the model results have been reported in [1].[2]

While making the measurements it was observed that typical service times for most input/output accesses to the disk subsystem were in the range of 35-45 ms. However, rollout accesses had typical service times in the range of 150-250 ms. Input/output software for CDC processors normally processes these two kinds of access identically, i.e., a new file is allocated space in the next (or most) available spindle, regardless of the type of the file. Thus the disk subsystem appears as a number of identical servers, with a service time distribution containing two distinct components with widely differing means.

When the two new CDC Cyber 170/750 processors were installed at UTA, the disk subsystem consisted of 12 spindles and three controllers. After examining the preliminary measurement results, it was conjectured that the merging of rollout file accesses with regular file accesses was operating to the detriment of the overall performance of the disk subsystem. When the rollout files were confined to a single disk spindle, connected to the processors through a fourth controller, a spectacular improvement was observed, in that the interactive response time for short requests dropped by a factor of four.

As it was not clear whether the improvement was due to the separation of the two file types, or to the additional capacity provided by the new controller, a study was instituted to examine various disk subsystem configurations, and recommend a best configuration for maximizing the performance of the overall system. It has been found that separation of rollout files from regular files results in improved performance, except in certain cases of severe overload of the disk subsystem.

In section 2, the important characteristics of the disk subsystem are discussed. Section 3 contains a discussion of the experimental environment in which the disk parameters were measured. Section 4

---

[1] Present address:
Department of Computer Science
Concordia University
Montreal, Quebec, Canada

[2] Figures in brackets indicate the literature references at the end of this paper.

introduces the simulation model. Section 5 gives some data concerning the observed performance of the various parts of the disk subsystem. In section 6, the model projections for various 12-spindle configurations are presented, and the observed effects are explained. In sections 7 and 8, configurations with altered file management algorithms and increased numbers of spindles are evaluated, and the desirability of separating rollout and regular file accesses is shown to be even greater than for the cases reported in section 6. Section 9 concludes the paper with discussion of the results and indications of directions for further research.

2. Disk Subsystem Parameters

2.1 Disk Subsystem Configurations

In the CDC Cyber 170 system measured, each 844-41 (IBM 3330-equivalent) disk spindle is connected to two 7154 controllers, which are each connected, via a channel and a peripheral processor (PP), to both central processors.

2.2 Data Arrangement

CDC 844-41 disks are sectored; there are 24 sectors per recording surface, and 19 data recording surfaces per cylinder, making a total of 456 sectors per cylinder. Because the 7154 controller can buffer only one sector at a time, a PP cannot transfer a received sector to central memory and issue the read function for the next sector before it has passed the read heads. Therefore files are allocated to alternating sectors: a logical track consists entirely of even-numbered sectors on a cylinder, or odd-numbered sectors. This is called half-tracking by CDC. For the UT-2D control program [2] used on the system measured, a logical track contains 228 sectors. For CDC's Network Operating System (NOS) [3], running on CDC 6000 Series or Cyber processors, the disk is divided into top and bottom (9-surface) halves, and a logical track contains 107 sectors.

2.3 File Types

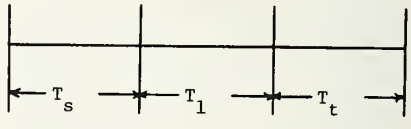
For UT-2D and NOS, files are of four types: system files, rollout files, permanent files, and local files. In UT-2D, permanent files are allocated using a ring counter, to spread them evenly across all spindles. Local files are allocated to the "least full" spindle. Space for all file types is allocated in units of one logical track. For UT-2D, when a logical track is requested to create or extend a file, the first (lowest numbered) track on the selected spindle is chosen. This tends to pack the disk towards the lower-numbered cylinders, and to spread all file types evenly across all in-use cylinders. When the last sector of a logical track is read or written, the PP program must follow a track chain (in a table in central memory) to determine the location (on disk) of the logical track containing the next sequential sector of the file.

I/O software within NOS and UT-2D requests transfers in units of a sector, which makes it convenient to specify any unit of transfer from one sector up to several logical tracks in one I/O request. Requests originating from user programs normally involve the transfer of small quantities of information. Requests originating directly from control program activity normally involve the transfer of much larger blocks of information (e.g., a whole compiler), and are also much less frequent.

2.4 Disk Spindle Parameters

The elapsed time (service time), T_d, at a disk spindle required to complete a request for data transfer to/from a moveable-head disk consists of three components (see Figure 2-1):

1. the time required to move the reading head to the cylinder containing the data (seek time, T_s);
2. the time required to rotate the spindle so that the data to be transferred is under the read/write head (rotational latency time, T_l);
3. the time required to actually transfer (read/write) the data between central memory and the disk spindle (data transfer time, T_t).



T_s : seek time

T_l : rotational latency time

T_t : data transfer time

Figure 2-1: Disk Subsystem Service Times

The time required to complete an I/O request consists of the disk service time, plus waiting times for the disk, the controller, the channel, and a peripherial processor.

3. Experimental Environment

3.1 System Configuration

The main computational resource at the University of Texas at Austin Computation Center (at the time of the study) consisted of two Cyber 170/750 central processors, each with 262,144 60-bit words of central

memory, 20 peripheral processors, and 24 channels. The two Cyber processors share access to a 505,204 word extended core storage (ECS) facility, and to a mass storage (disk) subsystem. The disk subsystem consists of four CDC 7154 disk controllers and twelve spindles of CDC 844-41 disk storage.

The two processors run under control of the UT-2D operating system [2]. The normal mode of operation is that interactive jobs are run on CPU A, and batch jobs are run on CPU B. When interactive jobs request input from their terminal, they are rolled out to ECS. If ECS gets too full, jobs with long residence times are swapped to a disk spindle with its own controller. Batch jobs are rolled out directly to the same swapping spindle from central memory, without going through ECS. System, local, and permanent files are kept on the remaining eleven spindles, and accessed through three dual-access controllers.

UT-2D has two primary components in each processor: MTR (MoniToR), which executes in PP zero, and which is in overall control of the system (on that processor); and CMR (Central Memory Resident), which runs on the central processor, and performs functions which are difficult or inconvenient for MTR to do itself (usually because they require substantial computational resources). One other PP on each processor is dedicated to managing the system console display; the remaining 18 PPs form a pool to be used for input/output operations and for other operating system functions.

3.2 Data Recording

Data concerning the operation of UT-2D were recorded by an event-trace probe embedded in MTR and CMR [4], using a buffer managed by CMR. This buffer is periodically written to tape by a transient PP routine. For the models and data reported later in this paper, two tapes were running (one for each machine), over the same time period, but the event recording was not synchronized between the two machines.

Data concerning the disks were recorded at the conclusion of each data transfer, by the disk driver which is resident in all PP routines. All waiting times that occur from the time that the channel is requested to the time that the channel is released are recorded. If the data transfer involves multiple logical tracks, the data for each logical track are recorded separately, but it is possible to distinguish these cases.

The data are reduced by an event delogging program called EVENTD [5], which produces statistics and histograms for events of interest, as well as a summary page of overall statistics.

4. A Queueing Network Simulation Model

The results described in section 5 have been used in a simulation model of the configuration described in section 3.1. The model is written in a queueing network simulation language called QSIM [6]. The QSIM preprocessor converts the model description into a Fortran program. A simulation model was used because of the desire to accurately model the details of the seek, latency, and data transfer activity, and because of the two-path switches in front of both the disk spindles (access via two controllers), and the controllers (access from two CPUs).

The model has two levels, which represent the general job flow and the I/O subsystem, respectively. The first level consists of two closed chains which model the processors, the tape drives, the rollin/rollout servers, and the user think time. The second level model is patterned after the physical interconnections (among channels, controllers, and spindles) which occur in the actual system. Because of the fact that these resources are held simultaneously by a job, passive resources [7] are used heavily in the model. The complete model is described in Yu [8].

This model was validated by comparing its performance statistics to the observations reported in section 5. The essential feature required for model validation was the introduction of the multi-part model for the data transfer characteristic, as described in section 7.

5. Measured Values

5.1 Interactive Response Time

When an interactive transaction is completed, the program image is rolled out to ECS. If the program image resides too long in ECS, it is copied to disk. When the interactive user hits carriage return, those images which have spilled over to disk must first be copied back to ECS, prior to being rolled into central memory. The interactive response time thus consists of the time between the initiation of a request for rollin and the initiation of a request for rollout, for all transactions, plus the time required to copy from disk to ECS (the disk subsystem response time for a rollin request), for those transactions which have spilled over, averaged over all transactions.

5.2 Disk Subsystem Performance

Two trace tapes representing quite different load situations are presented in the two following subsections.

5.2.1 Trace Tapes 72/73

Trace tapes 72 and 73 were taken on a busy day, at about 11.30 a.m. The number of interactive users on line was 109, but the demands that were placed on the two processors were relatively light. Therefore these tapes represent a situation where no component of the disk subsystem is a bottleneck (i.e., has a substantial queue), so that the disk subsystem is, in some sense, operating in a "linear" range. Note, however, that the disk and controller utilizations are substantially in excess of

those recommended by CDC for this disk subsystem.

Table 5-1: Disk Subsystem Times--Tapes 72/73

	Interactive Pop*	Time	%	Batch Pop*	Time	%
Regular Accesses						
Chan wait	0.204	9.6	14	0.107	7.0	10
Cont wait	0.240	11.3	17	0.294	19.2	28
Spin wait	0.089	4.2	6	0.053	3.5	5
Spin serv	0.911	42.8	63	0.586	38.3	56
Total	1.444	67.9	100	1.040	68.0	100
Rollout Accesses						
Total		979.0			1064.0	

* Average number of jobs waiting or being served at each point.

Table 5-1 presents the components that make up the disk subsystem response time. For regular file accesses, actual disk service time accounts for two thirds of the disk subsystem response time, with the remaining one third resulting from queueing for a channel or a controller. Thus queueing is present at the channel and at the controller, but it is not severe. The last line of Table 5-1 gives the disk subsystem response time for interactive spillover rollout and batch rollout requests.

5.2.2 Trace Tapes 74/75

Trace tapes 74 and 75 were taken on the same day as trace tapes 72 and 73, but at 3.30 p.m. The number of interactive users increased only slightly to 112, but the demands that they placed on the processors were much heavier. Therefore the total controller capacity tends to limit the overall system performance, and the disk subsystem is operating at well in excess of maximum recomended utilization.

Table 5-2: Disk Subsystem Times--Tapes 74/75

	Interactive Pop*	Time	%	Batch Pop*	Time	%
Regular Accesses						
Chan wait	2.880	111.5	55	3.270	118.3	58
Cont wait	0.924	35.8	18	1.020	36.9	18
Spin wait	0.144	5.6	3	0.143	5.2	3
Spin serv	1.260	48.8	24	1.200	43.4	21
Total	5.208	201.7	100	5.633	203.8	100
Rollout Accesses						
Total		904.0			849.0	

* Average number of jobs waiting or being served at each point.

Table 5-2 presents the components that make up the disk subsystem response time. For regular file accesses, the controller wait is almost as large as the disk service time, and the channel waiting time is larger that the sum of all other wait times. Thus severe queueing is present at the channel and the controller, and the disk subsystem is overloaded. The last line of Table 5-2 gives the disk subsystem response time for interactive spillover rollout and batch rollout requests. These are actually slightly smaller than those for trace tapes 72/73.

6. Projections for Alternate Configurations

Several experiments have been performed to examine the question: what is the effect of merging rollout file accesses with regular (user) accesses to the disk subsystem? Since rollout transfers tend to be much longer than regular transfers, variance will be introduced into the service time for the disk spindles, which will have the effect of lengthening the response time of the disk subsystem for regular accesses. On the other hand, if the fourth controller (used for rollout files) is less utilized than the first three (used for regular files), then the average channel utilization for regular files may be lowered, resulting in a shortened disk subsystem response time. The disk subsystem response time for rollout file accesses should be shortened in all cases, because more paths will be available.

Three configurations of the disk subsystem are presented:

1. Three controllers and 11 spindles for regular file accesses, arranged in a triangle with (4, 4, 3) spindles on each leg, one controller and one spindle dedicated to rollout file transfers (configuration A). This is the actual hardware configuration measured.

2. Four controllers and 12 spindles, arranged in a square with three spindles on each leg. All 12 spindles are used indiscriminately for both kinds of transfers (configuration B).

3. Four controllers and 12 spindles, arranged as for configuration B, but with rollout files confined to a single spindle, and regular files on the remaining eleven (configuration C).

The motivation for configuration C is that it is "similar" to configuration A in the sense that only one spindle contains rollout files.

6.1 Trace Tapes 72/73

Table 6-1 reports the results for trace tapes 72 and 73. As can be seen, the rollout channel (channel 4, configuration A) has a utilization approximately equal to the utilization of the regular channels. Therefore, effects due to increased channel availability should be negligible, and the changes in system performance should be due entirely to the variance introduced by merging the two classes of files.

Examining the data for configuration B, it may be seen that the disk subsystem response time increases for regular accesses, and decreases for rollout accesses, as expected. The overall controller utilizations, channel utilizations, and logical channel queue lengths increase, for both the interactive and batch processors. As a result, the interactive response time increases by 13%, and the batch throughput drops by 2.4%.

Table 6-1: Utilizations and response times for tapes 72/73

Logical Channel Queue Length

Interactive					
1	2	3	4	Total	Config
.076	.080	.047	.430	0.633	A
.123	.156	.132	.117	0.528	B
.119	.059	.084	.822	1.084	C

Batch					
1	2	3	4	Total	Config
.035	.041	.028	.233	0.633	A
.101	.074	.083	.080	0.338	B
.081	.020	.041	.345	0.487	C

Channel Utilization

Interactive					
1	2	3	4	Total	Config
.415	.440	.384	.459	1.698	A
.468	.474	.467	.459	1.868	B
.598	.414	.361	.653	2.026	C

Batch					
1	2	3	4	Total	Config
.312	.332	.289	.415	1.348	A
.379	.376	.357	.361	1.473	B
.490	.287	.235	.560	1.572	C

Controller Utilization (interactive + batch)

1	2	3	4	Total	Config
.551	.575	.512	.607	2.245	A
.592	.599	.579	.579	2.349	B
.699	.531	.470	.756	2.456	C

Interactive Statistics

Configuration	A	B	C
Response time	554	626	695
Disk subsystem response time			
(regular)	67.6	92.6	103
(rollout)	979	417	897

Batch Statistics

Configuration	A	B	C
Throughput	25.4	24.8	24.0
Disk subsystem response time			
(regular)	67.9	96.6	104
(rollout)	1064	669	939

With configuration C, the controller and channel utilizations for those controllers connected to the rollout disk rise, while the others fall, resulting in an unbalanced situation, and a further decrease in system performance: interactive response time rises by 25% compared with configuration A, and batch throughput falls by 5.5%. Note that when the system becomes saturated, the two channels used for rollout files will bottleneck first because of the unbalance. This suggests that confining the rollout activity to a single spindle is undesirable, when regular and rollout files share controllers.

Table 6-2: Utilizations and response times for tapes 74/75

Logical Channel Queue Length

Interactive					
1	2	3	4	Total	Config
1.07	1.06	.757	.436	3.323	A
.661	.537	.484	.581	2.263	B
.422	.285	.476	1.42	2.603	C

Batch					
1	2	3	4	Total	Config
1.26	1.21	.800	.210	3.480	A
1.00	1.03	.951	.955	3.936	B
1.02	.627	.948	1.45	4.045	C

Channel Utilization

Interactive					
1	2	3	4	Total	Config
.771	.796	.762	.477	2.806	A
.733	.753	.742	.720	2.948	B
.806	.659	.638	.822	2.925	C

Batch					
1	2	3	4	Total	Config
.775	.823	.768	.413	2.779	A
.715	.723	.712	.693	2.843	B
.792	.623	.584	.795	2.794	C

Controller Utilization (interactive + batch)

1	2	3	4	Total	Config
.914	.930	.908	.629	3.381	A
.879	.890	.883	.866	3.518	B
.915	.826	.794	.920	3.455	C

Interactive Statistics

Configuration	A	B	C
Response time	1664	1505	1534
Disk subsystem response time			
(regular)	202	179	172
(rollout)	904	577	1033

Batch Statistics

Configuration	A	B	C
Throughput	42.3	38.8	38.8
Disk subsystem response time			
(regular)	204	244	242
(rollout)	849	740	947

6.2 Trace Tapes 74/75

Table 6-2 presents the results of the analysis of configurations A, B, and C for trace tapes 74 and 75. As the utilization of the rollout channel in configuration A is considerably smaller than the average utilization of the regular channels, spreading the files evenly over four controllers lowers the average channel utilization for those channels which were used for regular files in configuration A.

On the interactive processor, the decrease in response time due to more channel availability is sufficient to overcome the increase caused by the introduction of variance, resulting in a modest improvement in the response time (9.0% for configuration B, 7.8% for configuration C), and in the disk subsystem response time for regular file accesses. However, due to the larger variance on the batch processor (rollout file accesses are longer and regular file accesses are shorter), the net result is an 8.3% decrease in batch throughput, accompanied by an increase in the disk subsystem response time for regular file accesses. Note also that, because of controller saturation, the disk subsystem response time for batch rollout files accesses is worse in configuration C than it is in configuration A.

6.3 Discussion

These three experiments at two load levels appear to support our contention that for normal to heavy loads, the variance introduced by merging long and short accesses results in a reduced system capacity, while for extremely heavy loads the lowered channel utilizations compensate for the variance effects, making the separate controller configuration less desirable.

7. Projections for Altered File Management

The distribution of data transfer times for regular file accesses in UT-2D can be divided into four distinct components [1]:

1. An exponentially distributed component, with a mean of approximately 10 ms., representing normal user I/O;

2. An uniformly distributed component, with a mean of 158 ms., representing searches for end-of-information when updating random access files;

3. A component representing accesses requiring two seeks (i.e., the total data to be transferred in one access is split between two physical cylinders);

4. A component representing loading or checkpointing operations (typically requiring three or more seeks).

The second component is an artifact of the organization of UT-2D file accesses, and is necessitated by the fact that the position of the end-of-information sector within a logical track can only be determined by reading sectors from the logical track until end-of-information is discovered. This necessitates a search of, on average, one half of a cylinder, requiring one half of 19 rotations of the spindle, or 158.333 ms. In this section a series of experiments are reported in which this component is deleted (i.e., the required information is assumed to be stored in central memory).

Table 7-1 shows the change in disk subsystem response time for regular file accesses, for trace tapes 72/73 and 74/75. Comparing case I with case II, it may be seen that a substantial improvement in the disk spindle service time, T_d, has occurred. This drop in T_d is reflected in an even greater drop in the overall disk subsystem response time, T_r, due to reduction of queueing caused by high channel and controller utilizations. (For example, for trace tape 74, a drop of 7.0 ms. in T_d results in a drop of 90.0 ms. in the disk subsystem response time T_r.)

Table 7-1: Disk subsystem response times (regular file accesses).

Trace Tapes 72/73

Configuration			A	B		C	
System	case	T_d	T_r	T_r	chg+	T_r	chg+
Inter.	I*	42.8	67.6	96.2	+42%	104.	+53%
	II	33.4	45.1	82.0	+82%	84.3	+87%
Batch	I	38.3	67.9	96.6	+42%	104.	+53%
	II	31.8	44.6	78.4	+76%	81.8	+83%

Trace Tapes 74/75

Configuration			A	B		C	
System	case	T_d	T_r	T_r	chg+	T_r	chg+
Inter.	I*	48.9	202.	179.	-11%	172.	-15%
	II	41.9	112.	131.	+17%	138.	+23%
Batch	I	43.4	204.	244.	+20%	262.	+28%
	II	39.4	123.	177.	+44%	191.	+55%

*I : before deletion
II: after deletion
+percent change compared with config A

An additional result of interest is that, because the rollout file accesses and the regular file accesses now have mean service times that are even more disparate, the variance introduced by merging the two file types together causes a greater (percentage) degradation in the performance of the disk subsystem in case II than is observed for case I.

Given that the uniform component in case I is an artifact of the organization of file accesses in UT-2D, which may be easily corrected by reprogramming (in fact, this change has already been made), it follows that case II is the preferred mode of operation. This implies that separation of

rollout files from regular files is likely to be desirable, even if the rollout controller is underutilized. This result should therefore be applicable to other CDC operating systems, as the end-of-information position is stored in central memory (in the File Status Table entry) for systems such as CDC's NOS [3].

8. A 20-spindle Configuration

As the interactive processor utilization is less than 0.5 for a 12-spindle disk subsystem, and as other studies had indicated that interactive activity would saturate the disk subsystem at about 150 users [1], various 20-spindle configurations were investigated, with an interactive load of 200 users.

Various configurations were investigated: Configuration D is identical to configuration A, except that two controllers are added, with eight spindles between them. Configurations E and F arrange the 20 spindles into two triangles, with one rollout disk for each class of rollout file (batch and spillover interactive). Configurations G and H are similar to configuration C, except that eight spindles are added between two controllers, and two spindles are dedicated to rollout files, as for configurations E and F.

Table 8-1: Statistics for 20-spindle configuration.

Configuration	D	E	F	G	H	D
Response time	1854	1836	1744	1795	1813	1424
Throughput	46.6	43.2	45.1	43.1	44.1	47.2
Disk Subsystem response time						
Interactive						
(regular)	133	199	189	197	202	146
(rollout)	5280	955	792	792	829	387
Batch						
(regular)	113	185	159	185	179	122
(rollout)	1561	908	771	855	855	405

The first five columns of Table 8-1 give various statistics for the different subsystem configurations. There is little to choose among them: the largest and smallest response times differ by only six percent. However, the disk subsystem response time for interactive rollout file accesses is over five seconds for configuration D. This implies that this configuration is limited by the capacity of the rollout channel. (Configurations E, F, G, and H have two rollout spindles; configuration D has only one.) To compensate for this, the model was run using a rollout transfer time that was approximately 40% smaller. (This corresponds to the use of a full-tracking disk with a controller capable of buffering more than one physical record at a time.) The last column of Table 8-1 gives the result for this run. The interactive response time and the batch throughput are clearly superior to all other configurations studied.

This again reinforces our previous conclusion that rollout file activity should be separated from regular input/output activity, especially if it can be given adequate transfer capacity.

9. Discussion

9.1 Applicability of the Data

The data observed are for a unique operating system, executing a computational workload which is almost entirely research- and student-oriented, i.e., which has little business data processing component. However, except for the way in which direct access updating is done, UT-2D is essentially identical to NOS as far as organization of the I/O subsystem is concerned. Therefore we feel that the data and the models presented here are also applicable to CDC 6000 Series and Cyber systems operating under NOS. Valid models for NOS systems could be obtained either by deleting the direct access component (see section 7) in the present models, or by running the models with data gathered directly from a NOS system [9].

9.2 Comparison with IBM Systems

The transfer time characteristic has several components, and the details of its shape were essential to achieving validation of the system model. The characteristic is radically different from published characteristics for IBM 3330 systems [10, 11], in spite of the fact that the physical equipment (disk spindle) is identical. This difference may be due to the fact that the I/O software of UT-2D (and NOS) organizes data accesses to the disk subsystem in quite different ways from the ways in which System/370 I/O Supervisors organize data accesses. This difference is so strong that Zahorjan [11] found no differences in his results when he varied the shape or the mean value of the transfer time distribution, whereas our results depend critically on these parameters.

The major point of difference comes from the decision to optimize locally [i.e., to reduce the total time for an individual transfer (especially a long transfer such as a rollout access)]. This makes CDC systems fundamentally different from IBM systems (even though the disk spindles are electrically identical). For CDC systems, the possibility of extremely long transfers will lead to a very large variance in the service time for data transfers, which will interfere severely with the ability of CDC disk subsystems to provide good service, because the waiting time for the controller is very strongly influenced by the variance of the service time for the disk spindles.

9.3 Study Results

In section 6.3 we have shown that, except for extremely heavy loads, the measured system appears to operate more

efficiently if the rollout activity is separated from the regular activity. We note that the load represented by trace tapes 74 and 75 is unusually high (it was created by reducing the number of disk spindles from 20 to 12 for one day without informing the users), and as such would not normally occur in practice.

In section 7 we reinforce our conclusion, as the simulation model results project that operation conditions closer to those in CDC's NOS show a clear preference for separation.

In section 8 we note that the gains to be made can also be invalidated if the rollout path is not allocated sufficient transfer capacity. We therefore feel that it is necessary to separate the rollout activity from regular I/O, in order to maximize the performance of the disk subsystem, except when the whole disk subsystem is sufficiently overloaded that increased channel availability can overcome the effects of the variance introduced.

9.4 Future Work

A considerable amount of data is available in the trace tapes, which has not been used in the present study. Since the original study was done, a simulation model has been developed which was used to evaluate the utility of seek/read-write overlap in this system. The results are reported in [1]. Further refinement of the rollin/rollout model elements is planned. One solution to the long holding times is to apply preemption to the disk transfers, i.e., to break them up into multiple transfers with release of the channel periodically [12]. This will lower the variance of the disk spindle service time, but increase the elapsed time required to complete long operations, as interfering disk arm movement may take place. A study is currently planned to determine whether the benefits which can be obtained outweigh the disadvantages accruing from the preemption.

In addition, the continuous holding of resources on the path between the central processor and the disk spindle is a major factor contributing to the results presented in this paper. Another study will examine the performance improvement which results when the disk subsystem requests are centrally scheduled, i.e., when the required resources (PP, channel, controller, spindle) are not allocated until all are available.

The authors would like to acknowledge the assistance of the staff of the Computation Center at the University of Texas at Austin, expecially N. Ferson, W. Jones, Dr. T. Keller, G. Smith, and Dr. C. Warlick, and the encouragement of Dr. J.C. Browne, who invited the first author to visit the University of Texas during 1979-80. This study was supported in part by the Natural Sciences and Research Council of Canada, grant number A-8634, and in part by the Computation Center and the Department of Computer Sciences, University of Texas at Austin.

References

[1] Atwood, J.W., Yu, K.-C., and Mc Leod, A., An Emperical Study of a CDC 844-41 Disk Subsystem, Performance Evaluation (accepted for puplication).

[2] Howard, J.H., A Large Scale Dual Operating System, in Proceedings ACM Annual Conference 1973, pp. 242-248.

[3] NOS Version 1 Reference Manual, Control Data Corporation, St. Paul, Minn., 1978. Publication numbers: 60435400 (Volume 1), 60445300 (Volume 2).

[4] Howard, J.H., and Wedel, W.M., The UT-2D operating system event recorder, Technical Report TSN-37, University of Texas at Austin Computation Center, February, 1974.

[5] Howard, J.H., and Wedel, W.M., EVENTD-UT-2D event tape summary/dump, Technical Report CCSN-38 (Revised), University of Texas at Austin Computation Center, August, 1977.

[6] McGehearty, P.F., QSIM, an implementation of a language for the analysis of queueing network models, M.A. Thesis, University of Texas at Austin, Dept. of Computer Sciences, 1974. See also Foster, D.V., McGehearty, P.F., Sauer, C.H., and Waggoner, C.N., A Language for Analysis of Queueing Models, Fifth Annual Pittsburg Modeling and Simulation Conference, April 1974.

[7] Sauer, C.H. and MacNair, E.A., Simultaneous Resource Possession in Queueing Models of Computers, Performance Evaluation Review, vol. 7, no. 1, 1978, pp. 41-52.

[8] Yu, K.-C., The effect of system configuration and file placement on the performance of a CDC 844-41 disk subsystem, M.A. Thesis, University of Texas at Austin, Department of Computer Sciences, 1980.

[9] Lo Cicero, C., An event-trace study of the performance of the I/O subsystem for a CDC Cyber 172 computer, M.Comp.Sci Thesis, Concordia University, Dept. of Computer Science, 1980.

[10] Bard, Y., A Model of Shared DASD and Multipathing, Comm. ACM, Vol. 23, No. 10, October 1980, pp. 564-572.

[11] Zahorjan, J., Hume, J.N.P., and Sevcik, K.C., A Queueing Model of a Rotational Position Sensing Disk System, INFOR, Vol. 16, No. 3, October 1978, pp.199-216.

[12] Browne, J.C., The Interaction Of Operating Systems and Software Engineering, Proc. IEEE, Vol. 68, No. 9, September 1980, pp. 1045-1049.

CPEUG81

Performance Prediction and Optimization – Methods and Experiences

SESSION OVERVIEW
PERFORMANCE PREDICTION & OPTIMIZATION
- METHODS & EXPERIENCES

Thomas P. Giammo

Federal Computer Performance
Evaluation & Simulation Center

The two papers of this session entitled, "Tuning The Facilities Complex of UNIVAC 1100 OS" and "Approximate Evaluation of a Bubble Memory in a Transaction Driven System" are about as far apart in their basic nature as can be imagined while remaining within the context of the subject of this session. One is a highly pragmatic "how to" approach to obtaining specific operation system parameters in a particular operating system while the other is a highly abstract approach to predicting the general characteristics of a model of a computer system. In one sense, this diversity in content is a measure of a broad scope of our field. Each paper represents an important facet of the work that we need to perform in order to achieve results that are practical and capable of implementation as well as soundly-based from a theoretical viewpoint. I believe that each of the papers will be found to be very helpful to the audience which it addresses.

In another sense, however, the diversity in content is indicative of a continuing problem in our field -- the apparent difficulty of the theoretical and practical elements of the performance evaluation community in finding common ground.

TUNING THE FACILITIES COMPLEX OF UNIVAC 1100 OS

John C. Kelly

Datametrics Systems Corporation
Burke, VA 22015

Jerome W. Blaylock

System Development Corporation
Slidell, LA 70458

Performance of UNIVAC 1100 computer systems is largely determined by the way the operating system, 1100 OS, manages the hardware resources. Since workload and performance requirements are unique to a particular site, UNIVAC supplies an extensive set of performance parameters for tailoring the system. One set of parameters, those associated with the Facilities Complex are examined in detail. The Facilities Complex is responsible for managing mass storage (disks and drums). The parameters associated with the Facilities Complex are defined, performance hypotheses are stated, and sources of data for testing the hypotheses are discussed. This paper extends and updates work reported by Kelly and Route [4].

1. Introduction

The 1100 OS, commonly referred to as the EXEC, is divided into six major components: Symbiont Complex, Coarse Scheduler, Activity Control, Dynamic Allocator, Facilities Complex, and I/O Complex. The Symbiont Complex buffers data between I/O devices and main memory. The Coarse Scheduler selects which runs should be opened next. Activity Control manages CPU dispatching and allocates CPU quanta. The Dynamic Allocator manages main memory. The Facilities Complex manages peripheral storage devices and allocates file space. The I/O Complex controls all I/O operations performed on the system.

The Facilities Complex consists of a set of non-resident EXEC segments that must be loaded when their services are required. Services performed by the Facilities Complex include such things as file assignment, space allocation, and file cataloguing. A list of the frequently used segments is presented in Table 1. To keep track of what devices are on the system and how much space is allocated to which files, the Facilities Complex makes use of an extensive set of control tables. They are summarized in Table 2. The relationship among the DLT, DAD, search item, and lead item is presented in Figure 1.

Upon receiving a new request for mass storage, the Facilities Complex first checks the EST to determine if sufficient space exists on the device type requested. If sufficient space is not available on the requested device type, it selects another device type according to the selection rules defined at system generation time. The Facilities Complex then accesses the LDUST and MBT to allocate the mass storage requested.

The Facilities Complex processes facilities requests in three passes. The first pass checks for the availability of the facilities being requested. If the requested

Table 1. Main Segments Associated with Facilities Complex

Segment Name	Description
FIMAIN	The entry point for the Facilities Complex. All requests initially go to FIMAIN which activates subordinate segments to do the work.
FIASG	Process @ASG and @CAT requests.
FIFREE	Process @FREE requests.
FIUMQ	Process @USE, @MODE, and @QUAL requests.
FALL	Allocate mass storage in track (TRK) or position (POS) granularity.
FASEC	Allocate mass storage sectors to EXEC.
FIBRKI	Break facility request into a standard format for processing.
FICKAC	Check availability of communications line.
FICKAF	Check availability of mass storage space.
FICKAT	Check availability of tape drives.
FICRIT	Create PCT item for file being assigned.
FIHOLD	Place a run in a facilities hold state (put on HOLDQ) due to unavailability of facilities.
FREL	Release mass storage.
FRSEC	Release mass storage sectors assigned to EXEC.
ROLBAK	Initialize a canned run to reload a file.
ROLOUT	Initialize a canned run to unload a set of files.
DRC	Manage the master file directory.

Table 2. Control Tables Used by Facilities Complex

Control Table	Description
MCT	Master Configuration Table. Describes the attributes for all devices configured on the system.
MFD	Master File Directory. Describes the attributes of all catalogued files in the system
MBT	Master Bit Table. Defines how space is allocated on each mass storage device.
DAD	Device Area Descriptor. Describes each contiguous block of mass storage allocated to a file.
DLT	Directory Lookup Table. Part of the MFD that contains a pointer for each catalogued file to either a search item or lead item.
EST	Equipment Summary Table (also called EQPSUM). Defines the number of positions and tracks available on all mass storage devices of the same type. There is one entry for each device type such as 8450's.
LDUST	Logical Device Unit Status Table. Defines the status of each mass storage device configured on the system.

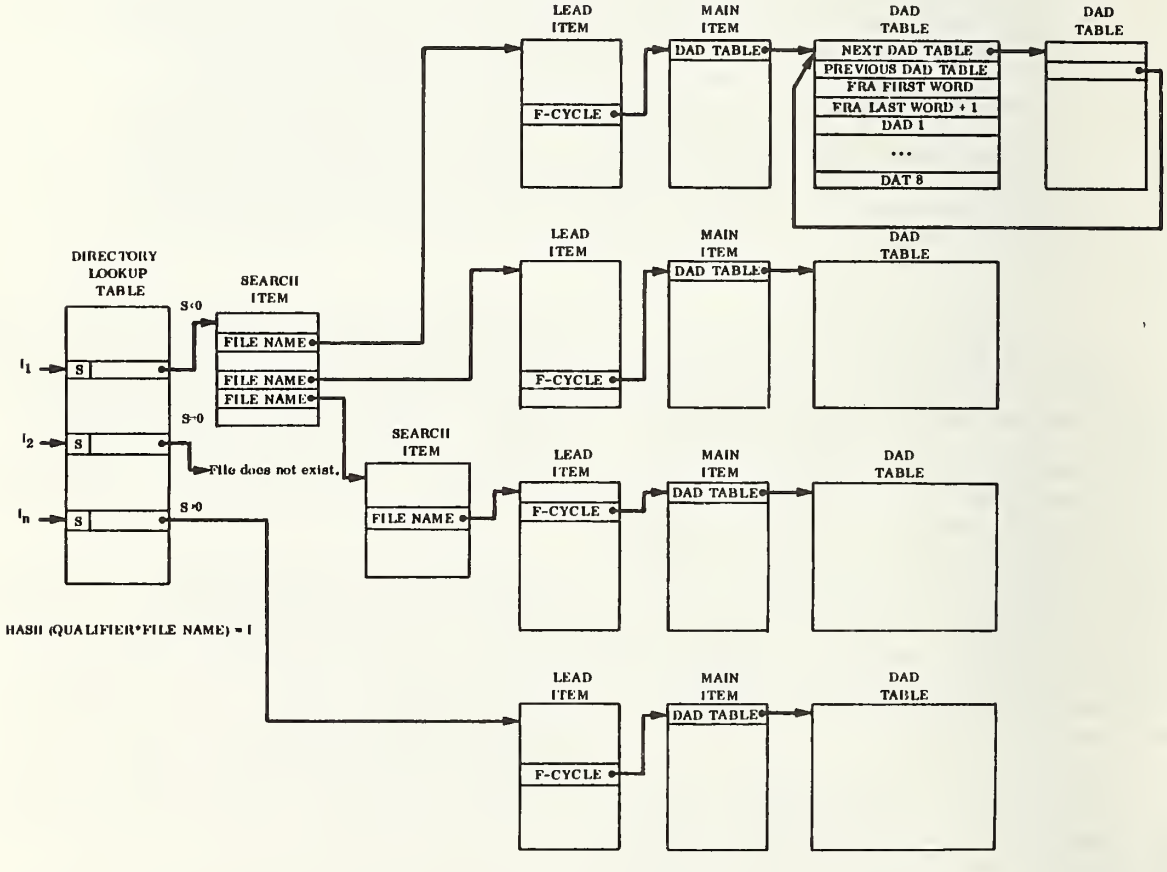


Figure 1. Master File Directory

facilities are not available, the run is placed on the facilities hold queue (HOLDQ) until they become available. When it is determined that all system resources requested by the run are available, pass one is completed. The second pass builds the required control tables and creates the MFD entries. The third pass performs the actual allocations. Only mass storage requests with initial reserves require allocation; if no initial reserves are specified, the allocation is handled dynamically by the I/O Complex.

The Facilities Complex has been studied in depth by [1,3,6] and as parts of larger studies by [2,4,5].

2. Virtual Mass Storage

Mass storage (peripheral storage on drums and disks) is managed as a virtual resource on UNIVAC 1100 systems. That is, from a user's viewpoint, the system contains an infinite amount of mass storage space. When more space is allocated than is physically available, the system creates space by rolling out infrequently used files to tape. When the ratio of allocated space to configured space becomes large, there is a great potential for the system to begin thrashing by moving files back and forth between real mass storage (disks) and virtual mass storage (tapes). The most severe problem occurs when a terminal user requests a catalogued file which has previously been rolled out to tape. The user must wait for the appropriate tape to be mounted and the file rolled back to disk -- a process which in extreme cases may require up to two hours [3,5,6].

The amount of mass storage available is monitored by the element FALL. When mass storage available drops below a specified limit, FALL calls ROLOUT to initiate the rollout process. ROLOUT starts a critical deadline batch run which calls the SECURE processor to perform the actual rollout. The SECURE processor is a utility program supplied by UNIVAC to maintain backup copies of all catalogued files. When started by ROLOUT, SECURE will unload enough files to bring mass storage availability up to a predefined level.

2.1 Mass Storage Thresholds

A set of user definable thresholds determine when the system should consider mass storage to be approaching saturation. When the available mass storage drops below threshold MSW1, mass storage is said to be "tight." This condition is detected by FALL which in turn calls ROLOUT to initiate the rollout operation. The criticality of mass storage is tracked by four other thresholds, MSW2 through MSW5. These thresholds trigger different throttling mechanisms to prevent available mass storage from being totally exhausted before rollout can be completed. The thresholds are defined in Table 3.

Table 3. Mass Storage Thresholds

Name	Definition	Default Value[2]	Action
MSW1	MSW2 + MSW4	3120	If mass storage available is less than MSW1, FALL calls ROLOUT. Also the minimum number of tracks to be unloaded.
MSW2	MSW3 + MSW4	2340	Print warning message on console.
MSW3	0150 * MAXOPN[1]	1560	Stop run scheduling with "CS H."
MSW4	MSW3 / 2	780	Print warning message on console.
MSW5	0200	128	Stop all requests except those by EXEC.

[1] MAXOPN is the maximum number of batch jobs allowed open at the same time.

[2] Default values are based on a MAXOPN value of 15.

Arguments can be made for setting MSW1 high or low. A low setting minimizes the average amount of unused (available) mass storage, while running the risk of reaching the critical thresholds where throughput will be curtailed. High settings reduce the risk of reaching the critical thresholds, but increase the amount of mass storage that remains unused. The proper setting really depends on the rate at which mass storage is consumed compared to the rate at which it can be made available with a rollout. The settings are also dependent upon the number of files with current backups when a rollout takes place.

2.2 Number of Tracks to Unload

The element ROLOUT calculates the number of tracks to unload from the formula:

$$NTRKS = \frac{TOTRKS}{PERCNT} - AVAIL \qquad (1)$$

where

TOTRKS = Total number of mass storage tracks configured on the system.
PERCNT = Fraction of configured tracks to be unloaded. (Current default is 10.)
AVAIL = Number of tracks currently available.

The number of tracks to be unloaded during a rollout varies depending upon the number of tracks available at the time the rollout is initiated. The objective is to maintain the number of available tracks approximately equal to PERCNT percent of the total tracks available.

The number of tracks to be unloaded is further constrained by the equation

$$MSW1 \leq NTRKS \leq TFAST \qquad (2)$$

That is, at least MSW1 tracks will always be unloaded, but never more than TFAST. TFAST is currently coded at 30,000 octal or 12,288 decimal and is historically based on the number of tracks on one FASTRAND. The number of tracks to be unloaded has important performance implications since it affects how frequently rollouts will have to be performed.

2.3 Selecting Which Files to Unload

The Unload Eligibility Factor (UEF) determines which files to unload during a rollout operation. If the UEF is improperly set, the wrong files may be rolled out resulting in an unnecessarily high number of rollbacks. Ideally, only infrequently used files should be rolled out. When a rollout is initiated, the SECURE processor is started via a canned run stream. SECURE computes the UEF for each catalogued file and selects those files with the highest UEF for unloading.

The UEF is computed from the formula:

$$UEF = (CRNCY*F(TSLR) + FREQCY*F(MTBR) + SIZEWT*G(SIZE))/2^{UEFWT} + EQBIAS + PVBIAS + FCBIAS + UEFBIAS \qquad (3)$$

where

- CRNCY = Currency weighting factor (default = 40).
- TSLR = Time since last reference in hours.
- FREQCY = Frequency weighting factor (default = 20).
- MTBR = Mean time between references in hours.
- SIZEWT = Size weighting factor (default = 10).
- SIZE = Size in tracks.
- UEFWT = Shift factor to reduce UEF magnitude (default = 5).
- F(X) = Function for computing point values for TSLR and MTBR.
- G(X) = Function for computing point values for SIZE.
- EQBIAS = Equipment bias (default = 2).
- PVBIAS = Private file bias (default = 1).
- FCBIAS = F-cycle bias (default = 1).
- UEFBIAS = User defined bias.

The functions F(X) and G(X) use Tables 4 and 5 respectively to compute point values that lie in the range 0-32.

Table 4. Point Values for TSLR and MTBR

POINTS	HOURS	DAYS	POINTS	HOURS	DAYS
1	1	.04	17	98	4.08
2	2	.08	18	116	4.83
3	3	.13	19	134	5.58
4	4	.17	20	152	6.33
5	7	.29	21	194	8.08
6	10	.42	22	236	9.83
7	13	.54	23	272	11.33
8	16	.67	24	320	13.33
9	22	.92	25	530	22.08
10	28	1.17	26	740	30.83
11	34	1.42	27	950	39.58
12	40	1.67	28	1160	48.33
13	50	2.08	29	1370	57.08
14	60	2.50	30	1580	65.83
15	70	2.92	31	1790	74.58
16	80	3.33	32	2000	83.33

Table 5. Point Values for SIZE

POINTS	TRACKS	POINTS	TRACKS
1	8	17	152
2	16	18	176
3	24	19	200
4	32	20	224
5	40	21	248
6	48	22	272
7	56	23	296
8	64	24	320
9	72	25	376
10	80	26	432
11	88	27	488
12	96	28	544
13	104	29	664
14	112	30	784
15	120	31	904
16	128	32	1024

Four types of files have predefined UEF's. They are:

- UEF = 0, Tape files or files unloaded
- UEF = 1, Removable disk files
- UEF = 2, Unload-inhibit files (G and V)
- UEF = 3, Empty files (0 tracks)

The computation of the UEF is illustrated by the following example.

```
Time since last reference  =  4 days
Time since last catalogued  =  104 days
Number of times assigned  =  46
Mean time between references = 104/46
                             = 2.3 days
Size of file  =  200 tracks
Stored on drum (EQBIAS)  =  2
Private file (PVBIAS)  =  1
Current F-cycle (FCBIAS)  =  1
UEFBIAS  =  0

UEF  =  (40*17 + 20*14 + 10*19) / 32
        + 2 + 1 + 1 + 0
     =  39.9
```

The information required to compute the UEF is maintained in the MFD.

3. Master File Directory (MFD)

The Master File Directory (MFD) is illustrated in Figure 1. The MFD describes each catalogued file in the system. Entries into the MFD are made by computing a hash code index from the file name:

```
HASH(QUALIFIER*FILENAME)  =  i        (4)
```

The hash code is used as an index into the Directory Lookup Table (DLT). The DLT then contains a pointer to a lead item which contains pointers, for each F-cycle of the file, to main items. The main item contains the descriptive information for the file. If two file names generate the same hash code index, a DLT conflict occurs. The conflict is resolved by replacing the lead item pointer with a pointer to a search item. The search item then contains the pointers to the lead items for the files with the duplicate hash codes.

3.1 Duplicate Hash Codes

Duplicate hash codes in the DLT result from (1) a DLT too small for the number of catalogued files, or (2) an inefficient hashing algorithm. The size of the DLT is specified by the system generation parameter DCLUTS. Besides defining the size of the DLT, DCLUTS is used in the computation of the hash code as follows:

```
HASH(QUALIFIER*FILENAME MOD(DCLUTS)    (5)
```

Analysis by UNIVAC and user sites [5] has shown that DCLUTS should be at least twice as large as the number of catalogued files. Since DCLUTS is also used in the computation of the hash code, it should also be a prime number. The use of a prime number improves the likelihood of generating hash codes that are uniformly distributed over the range of the DLT.

The actual hashing algorithm has been found to be effective when the above guidelines are followed. Analysis has shown that the algorithm is relatively insensitive to the use of similar file names. That is, similar file names do not usually hash into the same index. This is contrary to what was previously reported in [4].

Persing [5] analyzed the hashing algorithm using 7,357 file names. His findings are summarized in Table 6. Unfortunately, he did not give the actual values used for DCLUTS. His data are rounded to the nearest thousand. Two items from his study stand out. First, low values for DCLUTS result in nearly every file requiring a search item. Second, even when the size of DCLUTS is large, 40% of the files still required a search item. Even for a perfect hashing algorithm, there will always be some chance of filenames hashing into the same index. Duplicate entries in the DLT are undesirable for two reasons. First, an extra I/O is required to read the search item. Second, mass storage space is wasted storing search items. On release Level 36 of the EXEC, DCLUTS is set to a default value of 12,047.

Table 6. Results of Hashing 7,357 File Names

DCLUTS	RATIO TO NO. FILES	FILES REQ'ING SEARCH ITEMS	% OF TOT FILES
2,049	0.3	7,128	97%
17,000*	2.3	2,926	40%

* Approximate value rounded to nearest 1000

3.2 MFD Lock Cells

Access to the MFD is controlled by a set of lock words. Multiple lock words are used so that two users can access different parts of the MFD at the same time. The number of lock words and equivalently the number of parts of the MFD that can be accessed simultaneously is computed from the system

generation parameter DLOKSF. The number of lockwords is equal to

$$2^{DLOKSF} \qquad (6)$$

where DLOKSF is a positive integer between zero and four. As it turns out, the EXEC does not take full advantage of the multiple lock words. The dominate user of the MFD is the Facilities Complex, and the first thing that FIMAIN does is to set all MFD lock words. Since the Facilities Complex is designed to service only one request at a time, DLOKSF may as well be set to zero. When the Facilities Complex is rewritten to process multiple requests at the same time, it will be advantageous to increase DLOKSF to three or four.

4. File Granularity

Mass storage is allocated in either track or position granularity. A track is 64 sectors (1 sector = 28 words); a position is 64 tracks. The number and location of the granules assigned to a file are defined in the Device Area Descriptor (DAD) table. The DAD table defines every contiguous block of mass storage assigned to a file. The DAD table also maps a file relative address to a device relative address.

4.1 Mass Storage Fragmentation

Mass storage can become fragmented much the same as memory. When mass storage space is allocated to a file, the EXEC attempts to assign a contiguous block of space; however, if none is available, it will assign discontiguous areas to the file. Since each separate area of mass storage is defined by a separate DAD table, extra overhead is required to access that file. The overhead shows up as extra mass storage space required to store the DAD tables, extra memory space to hold the tables while the file is active, and extra CPU time to search the DAD tables.

A common cause of too many DAD tables for a single file is requesting no initial reserve for the file. When no initial reserve is requested, the system will allocate space in four track increments when it is needed. This relults in many discontiguous areas and multiple DAD tables. Some sites have minimized the likelihood of fragmentation by changing incremental allocations to sixteen tracks. Fragmentation can also be reduced by requesting position granularity rather than track granularity. Care must be taken, however, since position granularity can result in wasted mass storage space. It can also result in rollouts if no full positions are available.

4.2 DAD Table Storage in EXPOOL

EXPOOL is the EXEC's main memory storage pool. It is an area of main memory reserved for the exclusive use of the EXEC to hold such items as DAD tables. The parameter NGTB specifies the maximum number of DAD tables than can be in EXPOOL per catalogued file when EXPOOL is "tight." When EXPOOL reaches a critical threshold of utilization, access to EXPOOL is limited and one of those things limited is DAD tables. By keeping the number of DAD tables per file below NGTB (default value = 5), that is, minimizing mass storage fragmentation, this will not be a problem. When the value goes above five and EXPOOL is tight, extra overhead will result from moving DAD tables back and forth between main memory and mass storage.

4.3 Mass Storage Sectors

Each DAD table is one sector (28 words) in length. The parameter MAXSEC specifies the maximum number of DAD tables that the EXEC can read into memory at one time. If the number of DAD tables per file is low, MAXSEC may be low. A guideline in [4] suggests setting MAXSEC to the 80th percentile of the number of DAD tables per file. The default value is 124.

4.4 Maximum File Size

An option of the file assignment statements allows the user to specify the maximum file length in granules. When this value is unspecified, the parameter MAXGRN is used as a default value. If MAXGRN is set too high, mass storage may accidently be wasted by undebugged or inefficient programs. The current default value for MAXGRN is 128. As a guideline, Kelly [4] suggests that MAXGRN be set equal to the 80th percentile of all file sizes.

5. Performance Problems and Hypotheses

Potential performance problems associated with the Facilities Complex are summarized below. They are denoted as P1, P2, ..., etc. for ease of reference. Associated with each potential problem are a set of performance hypotheses denoted H1, H2, ..., etc. that might be the cause of the problem. Each hypothesis also has a brief statement of how to test the hypothesis.

P1. Too many rollouts are occurring.

H1. Not enough tracks are being unloaded. Monitor the frequency that rollouts are occurring. Verify that MSW1 is properly set. Monitor the frequency of the threshold messages on the console.

H2. Users are allowed to allocate mass storage files with no concern for the physical resources available. Monitor the number of files assigned to each user, their space requirements, and their frequency of use. Archive all files older than a specified number of days.

H3. File requests arrive in a burst pattern. Prepare a histogram of the interarrival time of rollouts to determine if time dependent conditions are causing a problem.

H4. Several very large files dominate mass storage. Sort files by size. Examine the usage pattern of large files. Consider removeable packs as an alternative.

H5. Insufficient mass storage is configured. Quantify the frequency of rollbacks and the time to complete a rollback. Compare to other sites. Compare to internal standards and past performance. Track the growth of the overcommitment ratio -- the ratio of mass storage space assigned to the mass storage space configured.

H6. Too many files have unload inhibited. Examine all G and V option files to determine if they are necessary. Track the growth of G and V option files as a percent of total files.

P2. Too many rollbacks are occurring.

H1. The wrong files are being rolled out. Check the UEF's of the files being rolled out. Verify that the weighting factors and points are set properly for your site. Establish guidelines to aid in setting weighting factors. Check all hypotheses for P1.

P3. The time to complete a rollback is too long.

H1. Heavy workload on the system. Since rollbacks run as critical deadline runs, this is unlikely; however, to verify the hypothesis, correlate the elapsed time of rollbacks with runs open and other workload measures.

H2. Many rollbacks queued up. Monitor the number of rollback requests waiting when a rollback run completes. Monitor the number of files rolled back per run. If more than one file is rolled back at a time, a queue must have existed.

H3. Long time to mount SECURE tape. Check operator console log or observe operators in action.

H4. Long tape search time. Observe rollback process in operation. Little can be done about this except limiting the number of files on a tape.

P4. The time to complete a rollout is too long.

H1. SECURE is doing physical rather than logical rollouts. When files have current copies on tape, they can be rolled out by simply making the space available. That constitutes a logical rollout. Physical rollouts occur only when no current backup exists. Physical rollouts should be avoided. Monitor rollouts to determine if physical rollouts are happening. Evaluate file save procedures to determine the need for backing up files more frequently.

H2. Too many files are being unloaded at once. Evaluate size weight in the UEF. Evaluate the mass storage thresholds. Monitor mass storage available.

H3. Heavy workload on the system. See H1 under P3.

P5. Mass storage availability remains high even though many rollouts are occurring.

H1. Too many tracks are being unloaded. See H1 under P1.

H2. The thresholds are set too high. Monitor mass storage available over time. Monitor frequency of threshold messages on operator's console. Determine rate at which mass storage is being used and adjust thresholds accordingly.

P6. System throughput is being held back by rollouts and rollbacks.

H1. Rollouts and rollbacks are happening too often. All of the analyses suggusted for P1 through P5 will be helpful in this area. Monitor the percent of runs that require rollbacks as well as the percent of time spent waiting for a rollback to complete.

H2. Excessive system resources are being used to manage mass storage. Compute the percent of system resources used by SECURE and other mass storage utilities.

Compare to resources used by user programs.

P7. A large amount of I/O is directed at the MFD.

H1. DCLUTS is set too low for the number of catalogued files. Compute the ratio of DCLUTS to the number of catalogued files. The ratio should be 2.0 or greater. Compute the percent of files requiring search items. Monitor percentage of search items on a regular basis.

H2. DCLUTS is not a prime number. DCLUTS should be a prime number at least twice as large as the number of catalogued files.

P8. Mass storage space is being inefficiently utilized.

H1. A large number of DAD tables are required per file. Prepare a histogram of the number of DAD tables per file. Compute average number of DAD tables per file. Check NGTB and MAXSEC for proper setting.

H2. Disk areas are being assigned but not used. Prepare a histogram of time since last reference for all files on the system. Pay special attention to all large files.

6. Data Analysis

Data on mass storage usage are available from several sources: LA, LASSO, DIRVER, and SECURE. The LA processor reads the Master Log File and produces both a ROLBAK and a ROLOUT summary report. The ROLBAK report lists the termination time of each ROLBAK run, as well as the number of files loaded, the average number of seconds per file loaded, the average number of tracks loaded per second, the number of tapes accessed, and the average number of files per tape. An identical report exists for ROLOUT's.

LASSO also summarizes data from the Master Log File; however, it does not provide a special report for rollouts/rollbacks. The information must be extracted from other reports under the ROLOUT/ROLBAK run ids. It is not nearly as comprehensive as LA.

DIRVER analyzes the MFD and produces a series of reports describing each catalogued file on the system. It provides excellent statistics on DAD tables, search items, lead items, and other attributes of the MFD. It also reports on the amount of mass storage space configured and available as well as many other mass storage usage statistics.

SECURE maintains a summary file from which several reports are available. These reports are useful in evaluating the variables used to compute UEF's. The SECURE Summary Report lists all the files unloaded, the time they were unloaded, the track size, and the UEF. The SECURE ROLOUT Report specifies the individual point values used to compute the UEF for rolled out files. Unfortunately, the UEF's for the rolled back files are unavailable. Since the time of the original roll out is not recorded, it is impossible to recalculate the UEF when the rollback takes place. This information is available by searching the SECURE Summary Report for the day when the file was rolled out.

7. Conclusion

Mass storage management, as performed by the Facilities Complex, has a significant impact on system performance. The article has emphasized tuning parameters; however, the performance analyst must not overlook the benefits to be gained by instituting a set of mass storage management quidelines. Many of the problems resulting from the virtual mass storage concept can be controlled or eliminated by exercising more stringent controls on how the users are allowed to assign mass storage. The approach used in this article has also been used to analyze other areas of the EXEC as described in [4].

References

[1] Bryant, R., A Performance Study of the Facilities Complex, Technical Papers: Spring USE Conference, March 5-9, 1979, pp. 29-38.

[2] Caldarale, C.R., State of Georgia Local Code, Technical Papers: Fall USE Conference, September 10-14, 1979, pp. 255-262.

[3] Gray, G., Mass Storage Files Management, Technical Papers: Spring USE Conference, March 5-9, 1979, pp. 381-398.

[4] Kelly, J.C. and Route, G.P., UNIVAC 1108 EXEC Level 32R2 Performance Handbook, NBS Special Publication 500-34, National Bureau of Standards, June 1978.

[5] Persing, T.D., Computer Performance Evaluation at USAF/AFSC/FTD, Technical Papers: Spring USE Conference, April 17-21, 1978, pp. 1-16.

[6] Richards, J. et al., A More Reliable File Maintenance System, Technical Papers: Spring USE Conference, April 17-21, 1978, pp. 55-198.

APPROXIMATE EVALUATION OF A BUBBLE MEMORY IN A TRANSACTION DRIVEN SYSTEM

Wen-Te K. Lin
Computer Corporation of America

and

Albert B. Tonik
Sperry Univac

An approximate queueing model is built to evaluate the performance of a transaction-driven computer system with three levels of memory hierarchy, one of which is the bubble memory. The criterion of performance is the average transaction response time excluding the time spent waiting for transaction initiation because it does not measure the system performance, but the amount of transactions entering the system. The results are presented in a form which can be used by the system users to decide how much memory to include in the system configuration to achieve desired transaction throughput rate and average response time.

1. Introduction

In a previous paper [1][1], we attempted to evaluate the effect of a virtual memory computer system with three levels of storage hierarchy by using a bubble memory rather than a head-per-track device (also referred to as a drum). From that evaluation we derived a mathematical model based on data collected from a virtual memory multiprogramming system which was frequently used by students to write their own programs rather than to run standard transactions. Therefore, they shared programs such as compilers and text editors. In such an application the limiting device was the intermediate storage level or paging device. Therefore, replacing the head-per-track device with a bubble memory increased system performance by almost the ratio of speed between those devices. Such an application reflects neither the sequential processing of large files, as in commercial applications, nor the terminals all accessing a large database, as in transaction oriented systems (real-time).

In this present evaluation, we are building a mathematical model for a transaction driven system. In such a system, the people at the terminals are entering transactions and waiting for replies. These transactions are processed against a common database. We obtained the specification for a standard transaction from data collected from Sperry Univac Transaction Interface Process installations.

The mathematical model attempts to evaluate the time to process a transaction, while processing "n" transactions simultaneously. This multiprogramming leads to queues at the various devices. The time spent waiting in each queue has to be added to the response time for each transaction. In addition, on some devices, as the queue gets longer, the throughput of the control units goes up, because of the ability to overlap latency periods. All of these interactions are built into the mathematical model.

2. The Standard Transaction

The transaction driven system we want to model is characterized as one with transactions entered from terminals. The trans-

(1) Figures in brackets indicate the literature references at the end of this paper.

actions are handled by standard programs and processed against a common database. The operating system occupies 100K words of core. The programs to process each transaction occupy about 5K words of core each. The processing of each transaction requires executing about 80K instructions. In addition, during the processing there are about 12 accesses to I/O devices. Of the 12 I/O commands, 8 would be for the secondary storage, and 4 for the mass storage. The CPU processing and I/O operations for a single transaction are performed in a sequential manner (no overlap between I/O and computing). Of course there is overlapping of I/O operations and computing during the processing of many transactions simultaneously.

The I/O is characterized as follows: on the average, an I/O operation will transfer about 250 words. About half of the I/O operations are reads and the other half are writes. However, of the writes about 10% are straight writes. The other 90% are writes where only part of the block is written. This operation entails a read followed by a write on the next revolution of the device.

In summary,

M_0 = 100K words
= resident software
M = the rest of the memory excluding M_0
m = 5K words
= program size for each transaction
I = 80K instructions
= cpu processing of each transaction
a_1 = 8
= no. of secondary I/O per transaction
a_2 = 4
= no. of mass storage I/O per transaction
$a_1 + a_2$ = 12 I/O
= no. of I/O per transaction
d = 250 words = 1 k bytes
= amount of data transferred per I/O
50% = percentage of read I/O per transaction
5% = percentage of straight write I/O per transaction
W = 45% = percentage of partial write I/O per transaction

3. The Hardware Configuration

The transaction system is assumed to run on a computer system with three levels of storage hierarchy. The second level memory are either drums or bubble memory (examples used will be Univac 8405E), the third level memory are movable-arm disks (examples will be Univac 8433E). We assumed the cpu is capable of at least 3 MIPs (million instructions per second). The revolution time of the drum is 16.7 ms. The revolution time and average seek time of the disks are 16.7 ms and 30 ms respectively. The channel data transfer rate is 900k bytes/sec. The latency time for the bubble memory is 0.8 ms, and the data transfer rate is 3M bytes/sec.

In summary,

t_1 = 16.7 or 0.8
= revolution or latency time for drum or bubble memory
t_2 = $1000/(9 \times 10^5$ or $3 \times 10^6)$
= channel time per drum or bubble I/O
t_3 = 30 ms + 8.35 ms
= average seek plus search time of the disk
t_4 = $1000/(9 \times 10^5)$
= channel time per disk I/O
t = 80K/3 MIPs
= cpu time per transaction

4. Mathematical Model

We can model the system as shown in Figure 1. Transactions come into the system

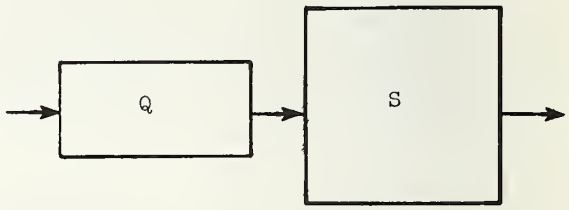


Figure 1. High-Level System Structure

from terminals and queue up at Q. The processor S then processes them. Let us assume transactions are generated by a Poisson process with an average rate of λ transactions per second. We also assume that system S can process up to N transactions simultaneously at the average rate of $\mu(n)$ transactions per second where $\mu(n)$ depends on the number n of total transactions in the system (being processed or in the queue). Therefore we have a birth and death process such that the birth rate λ is independent of the population size n, while the death rate is μ_n for $n \leq N$ and μ_N for $n>N$. Therefore we can calculate the stationary probability distribution Z_j that j transactions are in the system (including the ones being served) [2]. Let R represent the response time of a transaction which is measured from the time the transaction enters queue Q until the transaction exists from S, then

$$R = \sum_{k=1}^{\infty} \left(Z_k \cdot k \frac{k}{\mu_k} \right)$$

$$= \sum_{k=1}^{N-1} Z_k \cdot \frac{k}{\mu_k} + \sum_{k=N}^{\infty} Z_k \cdot \frac{k}{\mu_N} \qquad (1)$$

In order to calculate R, we have to calculate Z_k and μ_k. Let us assume:

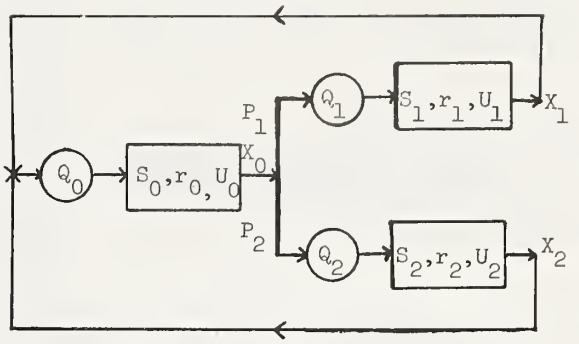
$$\Pi_o = 1, \quad \Pi_j = \frac{\lambda^j}{u_1 u_2 \cdots u_j} \quad j \geq 1 \qquad (2)$$

Then from birth and death process we know

$$Z_i = \frac{\Pi_j}{\sum_{k=0}^{\infty} \Pi_k} \qquad (3)$$

That means to compute R, all we have to do is to compute μ_k ($k \leq N$). In order to compute μ_k we have to look into the structure of the system S in more detail. The structure of S is shown in Figure 2. It is a closed queueing network with three stations. S_0 consists of the cpu and main memory, S_1 consists of secondary memory which can be either bubble memory or fixed-head disks, and S_2 is movable-arm disks. The size of the main memory excluding the part used by the operating system and any reentrant code is M. Since each transaction requires m bytes of main memory, the system can run at most M/m transactions simultaneously. Let us denote M/m by N. Each transaction, after being processed for an average of $1/r_0 = t/(a_1+a_2)$ seconds by the cpu, has the probability $P_1 = a_1/(a_1+a_2)$ of needing service from S_1, and the probability $P_2 = a_2/(a_1+a_2)$ of needing service from S_2. After being served by either S_1 or S_2, a transaction goes back to S_0 waiting for service by S_0 again.

We assume the service time in all three stations are exponentially distributed with average of $1/r_0$, $1/r_1$, $1/r_2$ respectively. Admittedly this is a rough approximation but it has been used by other people with good



Q_i = Queue length at station S_i (including the one being served) (per control unit at S_2)
S_i = Storage levels
$1/r_i$ = Average service time at S_i
X_i = Throughput of S_i (no. of operations per second)
P_i = Probability of a transaction going to S_i after being served by S_0
U_i = Utilization of S_i (fraction of time it is busy)
C = No. of control units at S_2

Figure 2. More Detailed System Structure

results. As outlined in a paper by Gecsei and Lukes [3], the following equations can be derived:

$$X_1 = P_1 X_0 \qquad (4)$$

$$X_2 = P_2 X_0 \qquad (5)$$

$$Q_0 + Q_1 + Q_2 = n \qquad (6)$$

$$U_0 = X_0 / r_0 \qquad (7)$$

$$U_1 = X_1 / r_1 \qquad (8)$$

$$U_2 = X_2 / C r_2 \qquad (9)$$

$$Q_0 = U_0/(1-U_0) \qquad (10)$$

$$Q_1 = U_1/(1-U_1) \qquad (11)$$

$$Q_2 = CU_2/(C-U_2) \qquad (12)$$

Here we have nine equations with nine variables: Q_i, X_i, U_i ($i = 0, 1, 2$), and 5 system parameters P_1, P_2, r_0, r_1, r_2. In order to solve the equations these five

parameters must be known. We have already shown that

$$P_1 = a_1/(a_1 + a_2) \quad (13)$$

$$P_2 = a_2/(a_1 + a_2) \quad (14)$$

$$1/r_0 = t/(a_1 + a_2) \quad (15)$$

Here equations for r_1 and r_2 will be derived. r_1 is the average number of operations system S_1 can process per second. It can be dependent on the queue length Q_1 if the queue is sorted according to sector positions. However, the definition of standard transaction states that about w of I/O are partial-write operation (a read is required before a write), this makes calculation of r_1 even more complicated. A formula for $1/r_1$ is formulated as follows, which has been validated by simulations (for detail of the derivation, see [4]).

$$\frac{1}{r_1} = \frac{[3(1+w)t_2Q_1+2(wt_1+t_2)+t_1]}{[2Q_1(1+w)]} \quad (16)$$

where w = ratio of I/O request that are partial write
t_1 = revolution or latency time for drum or bubble memory
t_2 = channel time without the extra revolution

r_2 is the average service rate of system S_2 which consists of a number of movable-arm disks and possibly more than one control unit. If there is no overlapping of seek operations among these disk drives, then r_2 is simply 1 divided by the average access time of the drive. But because of overlapping, r_2 is dependent upon the queue length Q_2. A formula for r_2 is derived as follows which has been validated by simulations (for detail derivation of the formula and its simulation validation results, see [5]). This assumes that all devices are on one control unit.

$$\frac{1}{r_2} = t_4 + t_3 \frac{\sum_{i=1}^{D} i^{(Q'-1)}}{D^{Q'}} \quad (17)$$

where t_3 = average seek plus search time
t_4 = channel time
D = no. of devices per control unit
Q' = queue length per control unit
$= Q_2/C$

In the following, equations (13) through (17) will be combined with equations (4) through (12) to derive Q_1 and Q_2. By eliminating Q_0, U_0, X_0, X_2, U_2, r_2 from equations (6), (10), (7), (5), (9), (12), and (17) we obtain

$$Q_1 = n - Q_2 - Y \quad (18)$$

$$Y = \frac{E^{Q'}C^2Q_2}{\left[t_4E^{Q'} + t_3\sum_{i=1}^{D} i^{(Q'-1)}\right] r_0P_2(C+Q_2) - E^{Q'}C^2}$$

where $E = \frac{D}{C}$

and $Q' = \frac{Q_2}{C}$

By eliminating Q_0, U_0, X_0, X_1, U_1, r_1, from equations (6), (10), (7), (4), (8), (11), and (16) we obtain

$$Q_2 = n - Q_1 - \nabla \quad (19)$$

$$\nabla = \frac{(1+w)2Q_1^2}{[3(1+w)t_2Q_1=2(wt_1+t_2)+t_1](1+Q_1)r_0P_1-(1+w)2Q_1^2}$$

Solving equations (18) and (19) for Q_1 and Q_2 to obtain closed form solution is very difficult. Therefore, we plotted curves for $Q_1 = f(Q_2)$ and $Q_2 = g(Q_1)$ in the Q_1, Q_2 plane for different values of n. Where the curves intersect is a solution for each value of n.

From the solutions for Q_1 and Q_2, we can obtain values for the other quantities. We can evaluate r_1 from Q_1 by using equation 16. We can find the r_2 from Q_2 by use of equation 17. Q_0 is obtained from equation 6.

We now want to obtain the response time to process a transaction once it has entered the memory.

$$R(n) = (a_1 + a_2)\, R_0(n) + a_1R_1(n) + a_2R_2(n) \quad (20)$$

Where $R_i(n)$ is the response time for each request to the ith facility when the multiprogramming level is n. The numbers a_1 and a_2 are obtained from standard transaction definition as defined in Section 3. The $R_i(n)$'s are given by Karlin, page 433 [2] as:

$$R_i(n) = \frac{1}{1-\frac{x_i}{r_i}} \cdot \frac{1}{r_i} \qquad (21)$$

From equation (7) through (12),

$$R_i(n) = (Q_i + 1)\frac{1}{r_i}$$

Now μ_k of equations (1), (2), (3), can be calculated as follows:

$$U_k = \begin{cases} \frac{K}{R(K)} & \text{if } K<N \\ \frac{N}{R(N)} & \text{OTHERWISE} \end{cases} \qquad (22)$$

By using μ_k's, we can compute the response time by using equation (1).

5. Results From Using Math Model

The results in Section 4 are applied to the transaction model and the system defined in Sections 2 and 3. Equation (20) is used to plot the internal response time for different multi-programming levels (which are related to the main memory size). This internal response time is the time required for a transaction to loop through the mass storage. It does not include the waiting time for the main memory, and it is directly related to the throughput of the system (through Equation 22). Equation (1) is used to plot average response time including time spent waiting for main memory, for various values of transaction input rate λ. The distribution of the response time can be obtained by using Equation (3). Three system configurations will be used. The first configuration consists of the cpu, one drum and six movable-arm disks controlled by a single control unit. The second configuration has two control units for the six disks.

The third configuration replaces the drum by bubble memory and has two control units for the six disks. The response times for these three configurations are plotted in Figure 3 and 4, Figure 5 and 6, and Figure 7 and 8 respectively. We can see the response time improves substantially from configuration 1 to configuration 2. By checking the

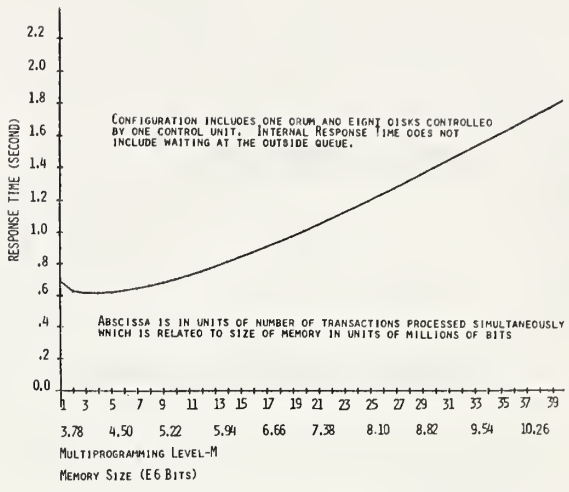


Figure 3. Average Internal Response Time R(M) vs. Multiprogramming Level M (Configuration One)

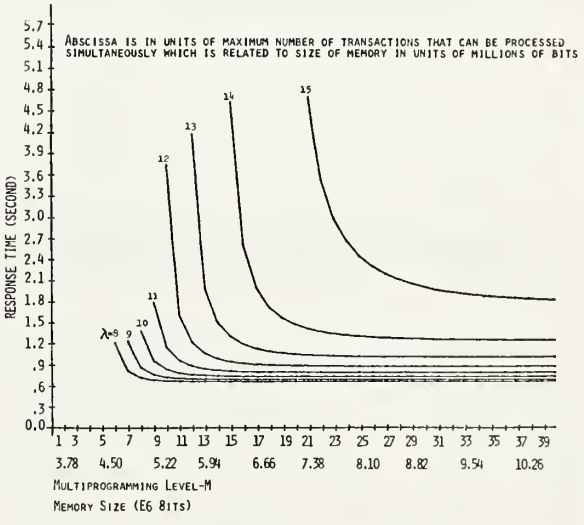


Figure 4. Average Response Time R vs. Multiprogramming Level M for Different Transaction Arrival Rate λ (Configuration One)

values for Q_0, Q_1, and Q_2, we find out it is because the control unit for the disks is the bottleneck. Therefore, doubling the number of disk control units improves the response time. But replacing the drum by bubble memory improves the response time only a little bit, not as much as the ratio of access time between drum and bubble memory, as shown in Figure 7 and 8. But if we increase the size of bubble memory, therefore reducing the number of I/O to the disks, the response time improves substantially.

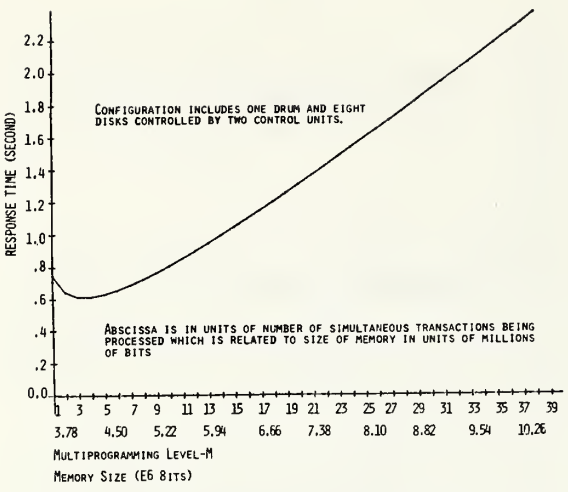


Figure 5. Average Internal Response Time R(M) vs. Multiprogramming Level M (Configuration Two)

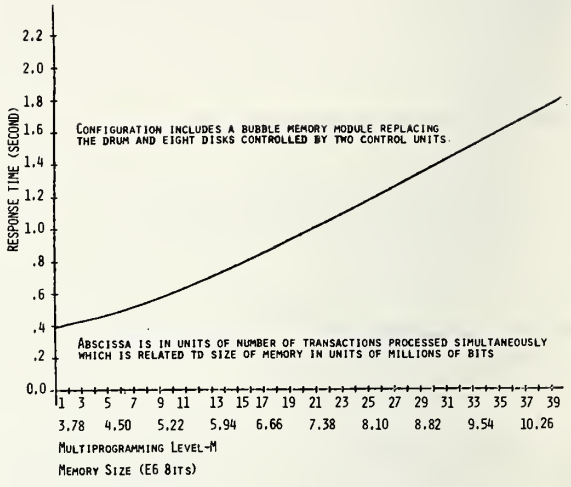


Figure 7. Average Internal Response Time R(M) vs. Multiprogramming Level M (Configuration Three)

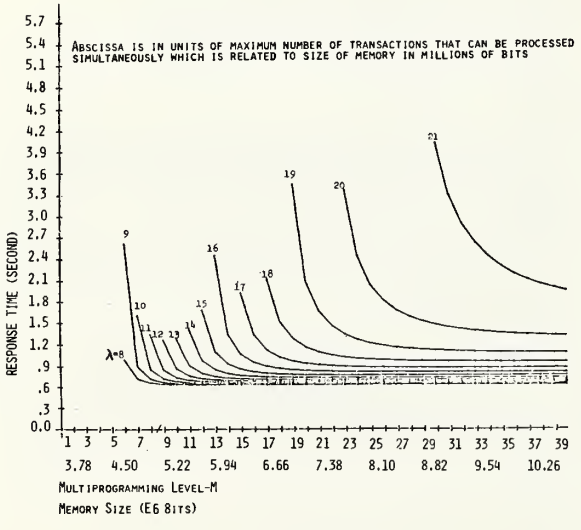


Figure 6. Average Response Time R vs. Multiprogramming Level M for Different Transaction Arrival Rate λ (Configuration Two)

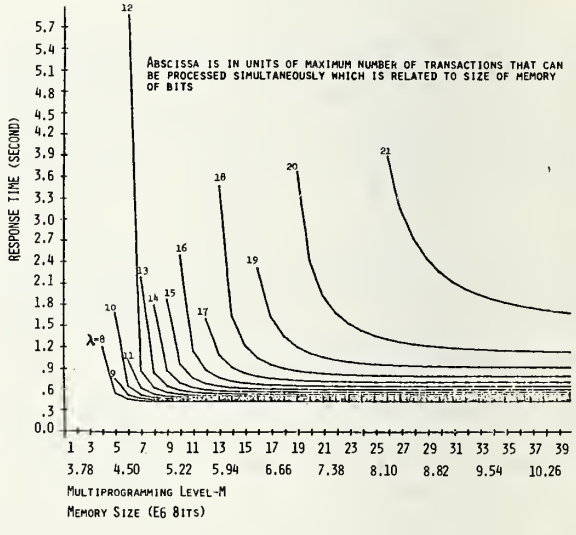


Figure 8. Average Response Time R vs. Multiprogramming Level M for Different Transaction Arrival Rate λ (Configuration Three)

6. Conclusion

A mathematical model for transaction processing system is built, which incorporates some elaborate algorithms for handling I/O at drum and disk memory. This model is then used to evaluate the effect of the bubble memory in a transaction system. The model is helpful when a user needs to decide a system configuration which can meet certain transaction rates with certain response times. It provides equations for plotting response time against both transaction rate and system configuration.

7. References

[1] Lin, W.T.K., and Tonik, A.B., Approximate Evaluation of the Effect of a Bubble Memory in a Virtual Memory System, Proceedings 13th Meeting of Computer Performance Evaluation -- Users Group, October 1977.

[2] Karlin, S., A First Course in Stochastic Processes, Academic Press, New York, 1969.

[3] Gecsei, J. and Lukes, J.A., A Model for the Evaluation of Storage Hierarchies, IBM Systems Journal, No. 2, 1974.

[4] Tonik, A.B., and Lin, W.T.K., Approximate Evaluation of the Effect of a Bubble Memory in a Transaction Driven System, Sperry UNIVAC Research Report, Blue Bell, Pa., 19406.

[6] Jewell, W.S., A Simple Proof of: L = w, Operation Research, 15, 1109-1119, 1967.

Communications Networks
-Technological Developments and Applications

SESSION OVERVIEW
COMMUNICATIONS NETWORKS
--TECHNOLOGICAL DEVELOPMENTS & APPLICATIONS

Jeffey Gee

Network Analysis Corporation
301 Tower Building
Vienna, VA 22180

Communications network is a critcal component in providing effective information services. Practically every business or industry today is involved in using some form of communication networks as a bridge between the people and the information resource. The recent echnology development in the area of long-haul and local area communication networks will greatly facilitate the information system user to pursue various organizational objectives such as resource sharing, increasing productivity and reducing cost. In order to effectively realize these objectives, a comprehensive network planning and design process must be achieved. The efforts should include communication requirements analysis, performance evaluation and network architecture modeling.

The objective of the session is to illustrate a broader prospective of communication networks. Consequently, the session is organized with an introduction of local network technology, a performance modeling of network node, and a network modeling case study.

The local area network has been and will continue to be one of the most important network tehnology developments. The first paper titled "Local Integrated Communications Systems: The Key to Future Productivity" presents an overview of a local communications network concept and describes its potential impact to organizational productivity. In communication networks, flow control traffic congestion avoidance is an important design consideration to meet performance requirements. An analytical model for evaluating the performance for a flow controlled network node is presented in the second paper. Finally, a case study presentation is given to illustrate the practical experience of modeling a long-haul star network.

LOCAL INTEGRATED COMMUNICATION SYSTEMS: THE KEY TO FUTURE PRODUCTIVITY

Andrew P. Snow

NETWORK ANALYSIS CORPORATION
301 Tower Building
Vienna, VA 22180

An overview of the types of local communication services required in typical large organizations is given. Such services as voice communication, data communications in support of Information Systems, Closed Circuit Television, Security Management, and Energy Management are discussed. Desirable characteristics of an integrated communications system that must provide a highway for these services are presented, with particular emphasis given to organizational productivity. Transmission media that are capable of forming the building blocks for an integrated communications system such as CATV, coaxial cable, microwave, fiber optics, and infrared/laser communication media are discussed in terms of the desired characteristics. Finally, an illustrative example of an integrated communication system is presented.

Keywords: CATV systems; coaxial cable; data communications; fiber optics information systems; integrated communications; local area networks, microwave; video teleconferencing.

1. Introduction

Distributed data processing (DDP) is adding great impetus to local area network developments. As computing and storage capabilities decentralize, the demand for intercommunication among more and more sophisticated users increases. DDP, in conjunction with integrated local communications systems, offers users a potential to improve individual productivity. This productivity enhancement is due not only to the new services that can be offered but also to the nature of the local network itself. This article will deal with local area networks as opposed to long haul networks. Long haul networks are intercity type networks where great emphasis is placed on bandwidth utilization and minimum distance layouts (as one pays for capacity by the airline mile). Local area networks are loosely characterized as being more intra and interbuilding oriented. For this reason they are typically on the order of a few kilometers. [1] In addition they are usually privately owned by the organization as opposed to leased facilities from a TELCO. Also they typically support data rates that far exceed the tariffed services or capacity of the TELCO facilities.

2. Services To Be Supported

In the intra and interbuilding environment, the local network should be able to support the wide variety of services that users are demanding. Typical data communications applications in buildings today are many terminals accessing one or more computer hosts for say database updating, query/response, and programmers engaged in systems development. In addition to data communications associated with some of the classical information systems,

the local network should also support the communications needs of security, fire, and energy management systems. These systems consist partly of many microprocessor based sensor devices that will typically report back to a computing/data base machine. In addition many security systems also have video monitoring requirements. Also, with the advent of video teleconferencing, modern corporate facilities will require video distribution systems. It is the video requirement that makes traditional twisted pair wire cable plant particularly unattractive in the future.

Another service area is being created by components of the "office of the future" Communicating word processing and document distribution machines are greatly impacting approaches to information flow in large organizations. Electronic mail services have already encouraged many executives to have terminals on their desks. In the near future, a terminal on every desk is not an inconceivable development in some organizations. The prolific increase of intelligent devices on the corporate scene is placing demands on classical building transmission plants that cannot be met in a cost effective manner.

3. Desirable Local Integrated Communication Media Attributes

In order to meet the diverse demands that organizations are and will be placing on building physical plants, consolidation is necessary in order to be cost effective. The principal attributes that must be considered in the selection of an integrated communications media are:

- flexibility
- connectivity
- capacity.

Each of these three attributes directly affect the ultimate productivity of the user organization. Productivity will be discussed in qualitative terms for each of these attributes.

3.1 Flexibility

Flexibility pertains to the ability of the integrated media to adapt to both tactical and strategic change. Tactical changes deal with the day to day exigencies of a viable organization while strategic changes deal with the evolutionary character of user demands. [2] Examples of tactical change might be the migratory nature of people/organizational units within a building or building complex, and concomitantly equipment migration. Typically, productivity during migratory periods is very poor for the organizational units involved. When people move many events are triggered. Terminals and phones move. Support personnel take word processing equipment with them. If the magnitude of the move is significant, simple adjustment of the building wire plant may not be technically feasible. The plant may not be able to support the proposed move. In the cases for which the plant can adjust, significant wire installation efforts are often necessary to get people within "reach" of the plant. Flexibility in this context means the provision of reasonable access of the media to the user's changes.

Strategic flexibility pertains more towards the ability to adapt to long term change without expensive architectural renovation to upgrade the media. In this context flexibility means inherent or expandable capacity to handle new requirements. An example of strategic flexibility would be the planned movement of a data center within an organization's building or building complex. With traditional wire plant, the data center is the hub of thousands of wire pairs that are distributed throughout a building. Movement of a data center within the same building might well cause organizational upheaval or expensive retrofit. In this case it is either extremely expensive or very unproductive as users are subjected to possible interruptions, degradations, or limitations on service during the move. A different example of strategic flexibility would be a planned information system which will require significant capacity in comparison to existing services. A media which could readily adapt to these two strategic migration examples would be desirable. Flexibility also means that the media should be prepared to accommodate very dissimilar types of communications (e.g., video and data).

3.2 Connectivity

Connectivity relates to the ability of the media to provide all required physical connections within the building or building complex. Traditional wire cable plants are point-to-point in nature. If a particular device requires connectivity to just a modest number of physically separated devices, the point-to-point approach can

become quite untenable. If wire is used, the central switching approach architectures with their significant reliability implications are required. Because of bandwidth limitations, wire can force the separation of media as opposed to integration. The any-to-any communications connectivity requirement within buildings is becoming more of a reality as time progresses. The trend is definitely towards the heterogeneous as opposed to homogeneous environment. The IEEE 802 committee efforts [3] on the standardization of both contention and token local area networks is encouraging the heterogeneous environment. This environment will in turn encourage the growth of multiple connectivity requirements on the corporate premises. Diverse services such as electronic mail, electronic document distribution systems, and communicating word processing systems will demand diverse connectivity within buildings. A strong case can be made for a broadcast type capability to insure any-to-any physical connectivity. The implementation of the any-to-any heterogeneous connectivity must await acceptance and pursuit of such standards as the International Standards Organization (ISO) open systems interconnection architecture. [4] Organizational productivity is closely related to this connectivity endeavor.

3.3 Capacity

Capacity means bandwidth necessary to support the tactical and strategic requirements of the user organization. Tactical capacity relates to the ability to provide the bandwidth necessary to support existing and short term requirements. Some of these bandwidth requirements might be of the point-to-point single destination type, while others might be of the multiple connectivity type. Besides data communications in support of information systems, video requirements may exist that require large amounts of bandwidth. Typical analog video requires approximately 6 mhz of bandwidth. Heavily utilized point-to-point requirements might be supported at the bandwidth required for the specific application. Others may be multiple connectivity or more bursty traffic types that might share high bandwidth channels of the contention or token nature. These channels typically will range from 1 to 10 mbps. Proper modulation techniques or bandlimiting techniques can limit a 10 mbps signal to 6 mhz on an RF type system or 20 mhz on a baseband system, respectively. The local integrated communication media should be able to accommodate all intra-building communications requirements. For this reason a large capacity is required, especially in comparison with twisted pair cable. If the system is not able to support these requirements, productivity can potentially suffer either because of disruptive expansions or because of the forced requirement of managing separate communications media.

4. Transmission Media Elements

The transmission media building blocks available in the construction of the integrated communications media are:

- twisted pair wire,
- microwave,
- fiber optics,
- coaxial cable,
- CATV cable systems,
- Infrared/laser.

Each of these media will be discussed and compared in terms of the flexibility, connectivity, and capacity attributes. The summary of this comparison is shown on Table 1.

	FLEXIBILITY	CONNECTIVITY	CAPACITY
Twisted Pair	Poor	Point-to-Point	Poor
Microwave	Fair	Point-to-Point	Good
Fiber Optics (Current Technology)	Fair	Point-to-Point; Limited Broadcast	Excellent
Coaxial Cable	Fair to Good	Point-to-Point; Broadcast	Excellent
CATV	Excellent -	Point-to-Point; Broadcast	Excellent
Infrared/Laser	Fair	Point-to-Point	Good

TABLE 1: INTEGRATED MEDIA BUILDING BLOCK

4.1 Twisted Pair

The characteristics of twisted pair have been alluded to in the previous discussion of each attribute. Wire is a point-to-point media with little broadcast potential. Moderate data rates must be supported by limited distance modems over distances that are beyond the capability of line drivers interfaces such as RS-232 or RS-449. Video capability is extremely limited

and specially conditioned pairs are necessary to pass video over very short distances. Wire is not a good candidate for an integrated media but it will continue to exist in buildings because

- it is there,
- it currently provides the best support for distribution of voice.

In the future, if CATV systems can support voice distribution in a cost effective manner, new buildings may potentially be void of twisted pair. Wire cable plant will most likely exist in parallel with whatever integrated media is suggested for existing buildings . It will become increasingly less attractive in the future.

4.2 Microwave

Microwave systems are point-to-point in nature. They would typically be used to link buildings together that are moderate distances apart or to bridge gaps for which the organization does not have easement rights. Microwave is also line of sight and is nominally limited to a 25 mile range. Of course the maximum range of this media cannot be used for extension of Carrier Sense Multiple Access Collision Detect (CSMA/CD) contention channels to other buildings because of propataion delay. Flexibility of microwave is closely related to capacity. Digital microwave links in the 90 mbps ranges are commercially available. For analog microwave systems 6 mhz passbands are typical. Use of microwave links requires FCC approval of frequency plans. The approval process sometimes requires frequency interference studies. Towers, line of sight requirements, and FCC approval make microwave inflexible at times as a media. Path reliability is a function of terrain, frequency of operation, path length, and atmospheric conditions, however extremely reliable paths are common.

4.3 Fiber Optics

Fiber optic systems, although principally point-to-point in nature, can be configured into a distribution system. This is done by splitters that subdivide the light energy into equal parts. The limitation of this continual splitting technique is that the power threshold required at receivers for acceptable bit error probability is rapidly achieved. Lightwave amplifiers are still R&D devices. No analog to the high impedance tap yet exists in the light world. Fiber optic broadcast distribution systems are probably five years away from common commercial application. The capacity of fiber optics is dramatic. Capacities in the range of several hundreds of mbps are possible. Flexibility of optics is only fair as Frequency Division Multiplex (FDM) techniques are still in the R&D stage. This means that very high data rates must be allocated in a Time Division Multiple Access (TDMA) scheme to effectively use this baseband datastream in other than a point-to-point fashion. Once the allocation scheme is fixed, changes are difficult. Also video support requires T2 (6.3 mbps) data rates for high quality video.

4.4 Coaxial Cable

Coaxial cable is distinguished from CATV in that CATV uses FDM techniques to create separate wideband channels while coaxial cable operates on a baseband basis. The coaxial cable system can be repeatered however and therefore its range is extendable. It suffers from the same flexibility problems mentioned in fiber optics distribution systems in that composite data rates are required to serve many distinct services. Some time slots may be dedicated while others might be used as a contention resource. Xerox's Ethernet (Trademark of Xerox Corporation) is a baseband coaxial cable system. Its capability to provide a user an integrated communications media is limited because of its baseband nature and inability to easily accommodate video. An advantage of this type of media is its inherent broadcast capability.

4.5 Community Antenna Television (CATV) Systems

CATV systems offer perhaps the best media for integrated communications within and among buildings in close proximity. Up to forty-one 6 mhz channels can easily be supported as shown on Figure 1. Channels can be used for contention systems, video distribution, and even point-to-point communications. As indicated in both [5] and [6], the service, architectural, and protocol local area network development issues are complex. CATV is the best suited all around local area network media. This media far surpasses the others in the flexibility attribute. This flexibility combined with the any-to-any physical connectivity and large bandwidth capability will make it the most commonly used integrated media type.

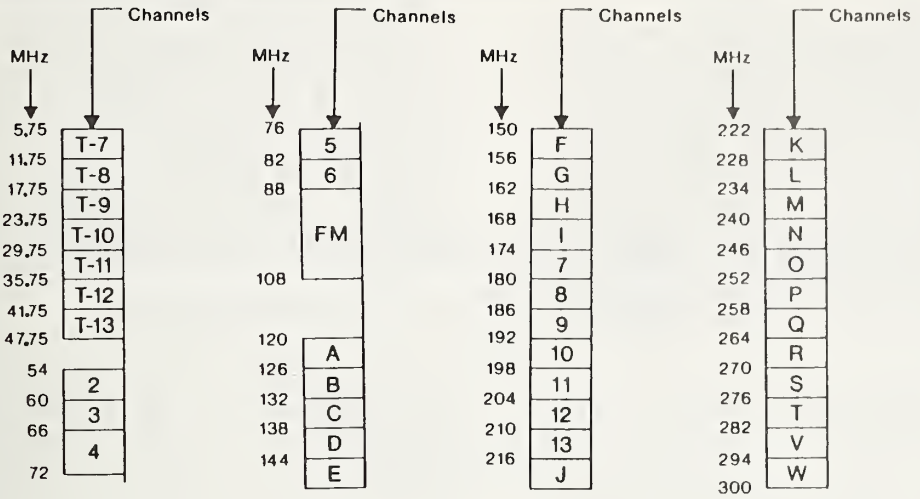


FIGURE 1: CATV FREQUENCY ALLOCATIONS

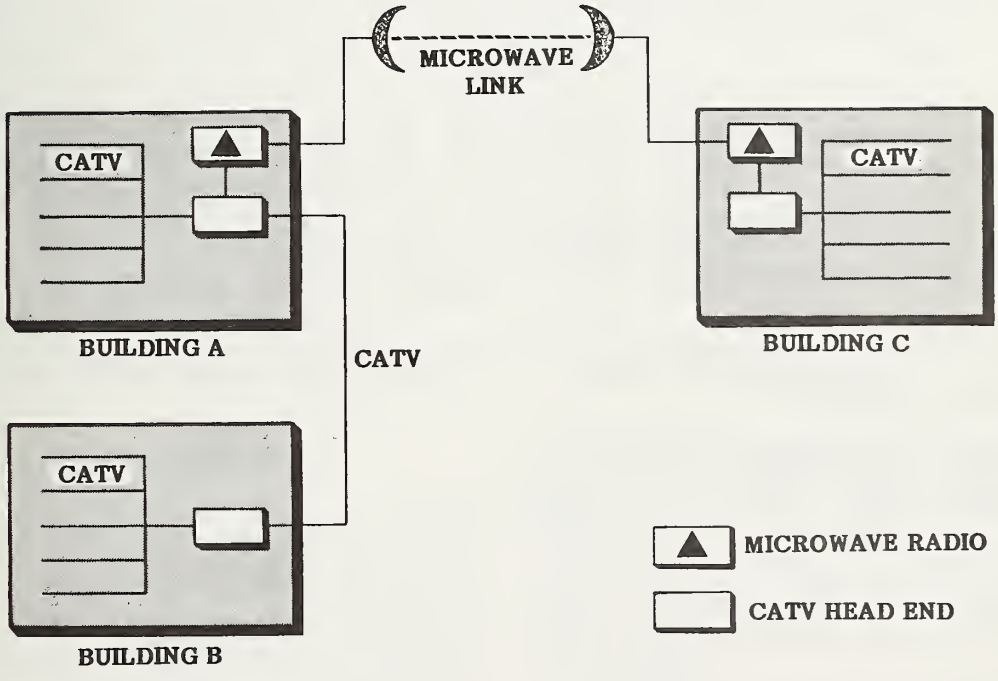


FIGURE 2: EXAMPLE OF A LOCAL INTEGRATED COMMUNICATIONS SYSTEM

4.6 Infrared/Laser

Infrared/Laser communications links are strictly point-to-point media. They are greatly limited in range due to fog or rain attenuation and any phenomenon that might diffract, disperse, or reflect the unguided but directed light energy. The flexibility of infared/laser links is about the same as microwave with the exception that desk top links through window glass are possible to establish short interbuilding links (as opposed to large tower or roof top mounted MW antennas).

5. Example of a Local Integrated Communication System

An example of a local integrated communications system is shown in Figure 2. In this example an organization spans three buildings. Each building has its own CATV distribution system for its own communities of interest. For common services, basic connectivity is provided by derivation of point-to-point circuits between buildings. In the first case connectivity between buildings A and B is obtained by extension of the CATV system. In the second case connectivity between buildings A and C is obtained by a point-to-point microwave system. This may have been required by lack of easement between the buildings. It would be possible to establish a common broadcast channel between buildings for use as a contention channel. This could operate on different frequencies in each building CATV system and not interfere with a wide assortment of other services on other CATV channels peculiar to each building. These services could include the entire suite of those mentioned in Section 3.

6. Conclusion

CATV systems offer the best media for local integrated communications systems. For organizational needs spanning several buildings these integrated systems will consist of such elements as fiber optics, microwave, and infrared/laser links. Wire plant will continue to exist in buildings but the appearance of buildings void of wire in the next five years is a real possibility. Baseband coaxial systems will also appear on the scene but are not good candidates for a flexible integrated system. CATV systems are the best all around system in terms of capacity, flexibility, and connectivity. These integrated local systems will profoundly affect productivity in the future.

References

[1] Tanenbaum, Andrew S, Computer Networks, Prentice Hall, 1981.

[2] Kaczmarek, R and McGregor P, "The Synthesis of Formal User Requirements: Defining the Networking Problem To Be Solved", IEEE 1978.

[3] IEEE 802 Local Network Standard, Draft 1, July 4, 1981, IEEE Computer Standards Committee.

[4] ISO/TC97, N537, Architectural Model of Open Systems Interconnection, December 1980.

[5] Maglaris, B and Lissack, T., "An Integrated Broadband Local Network Architecture", 5th Conference on Local Computer Networks, IEEE, October 6-7, 1980.

[6] Stack, T. and Dillencourt, K., "Protocols For Local Networks", Proceedings, Trends & Applications: 1980 Computer Network Protocols; May 29, 1980.

PERFORMANCE ANALYSIS OF A FLOW CONTROLLED COMMUNICATION NETWORK NODE

P.G. Harrison

Department of Computing
Imperial College
London SW7 2BZ
England

A Markov model is presented for a message processing node with batch arrivals, window size constraints and an additional flow control based on a credit allocation scheme to limit congestion effects. The model is sufficiently general for use in modelling many nodes which represent servers or sub-systems in any queueing network with congestion controls. Its principal application lies in the study of communication networks, in particular representing the network-independent flow control of messages between networks, which may have quite different message sizes and protocols, via a "gateway". No closed form analytic solution exists for the direct representation of the congestion control scheme, and the approach taken is to exploit its properties, approximately in some cases, to simplify the model, reducing the size of the state space sufficiently to permit direct, numerical solution of the balance equations. The credit allocation scheme is shown to be equivalent to the node's operation in different modes, each with its own buffer capacity. Great simplification is achieved in this way; credit control need no longer be modelled explicitly. The approximations made are suitably realistic to retain adequate representation of parameters, resulting in a good physical interpretation and accurate predictions. Accuracy is assessed by comparison with the results of explicit simulations of a selection of nodes of this type with various parameterizations. Finally, we suggest applications for the model in the assessment and comparison of performance under various congestion control schemes.

Key words: Communication networks; flow control; performance prediction; analytic models; queueing models; Markov processes; simulation; validation.

1. Introduction

In communication networks, flow control schemes are necessary to ease congestion which would otherwise cause any of a variety of problems. These include speed mis-matching (different processing rates at the sending and receiving nodes), deterioration in efficiency (e.g. throughput degradation), loss of security or reliability (e.g. due to buffer overflow at some node), unfairness (in the sense that certain classes of message may be allowed to dominate). Various strategies have been proposed and sub-sets of these implemented in some actual networks; e.g. pacing control, [6], window size constraint as in SNA, DECNET and ARPANET, [6], buffer size limitation at individual nodes, [8,9], and isarithmic control, [2]. Performance prediction of flow controlled networks is important in the comparison of such schemes, with the aim of choosing the most efficient (always subject to adequate attainment of the original objective of the flow control).

We address this via an analytic model, constructed initially to represent a node, the message processor, connecting separate communication networks with possibly quite

different characteristics, such as message size or protocol; a gateway. Flow control in the gateway is independent of the characteristics of each network and uses:

(i) Window size restriction;

(ii) Buffer size limitation via assignment of a threshold value for the number of messages therein;

(iii) Message acceptance protocol based on the availability and issue of credits, accumulated on completion of processing of a message unit by the message processor. This is a pacing control.

The analytic model was developed within a Markovian framework and results in balance equations with many global dependencies in the sense that certain state transition probabilities depend on several state vector component values; in particular, blocking is present. The credit-based control scheme is shown, in 2.1, to be equivalent to operation by the node in modes, representing congestion levels, each with its own buffer capacity. In this way, the problem specification is greatly simplified, and the state space much reduced, since credit handling need no longer be modelled explicitly. The approach taken is to make realistic approximations to further reduce the size of the state space, the number of dimensions in particular. The global state dependence, e.g. blocking, is still present, but the resulting small size of the state space permits direct numerical solution of its balance equations for the state space probabilities. From these, important performance measures follow immediately. The physical realism of the approximations made retains the original parameters of the model in an intuitively significant way, preserving an accurate representation. Thus the precision of predictions is expected to be good. Validation is performed by comparison with the results of simulation tests, although ultimately validation is only relevant when based on data measured on actual networks. Nevertheless, the consistency shown with the simulated results does increase confidence in the model; its approximations in particular.

The model has great generality in that it is the solution of raw balance equations, so that all its parameters can have any (global) state dependencies; in particular state dependent service rates. Thus it can be applied to a variety of queueing network nodes in the contexts of communications and computer systems. For example, whole sub-systems of physical nodes may be represented by one with state dependent service rate and/or customer acceptance protocol; indeed any system with a congestion control scheme which can somehow be represented realistically by a single (possibly very complex) processor is a suitable application.

In the following section, the gateway node modelled is defined and the analysis resulting in the reduced state space discussed. The balance equations for the underlying Markov Process then follow directly, from which the state transition probability matrix may be obtained. The validation exercise is described in section 3, and finally we suggest applications of the model and extensions which could be made to increase its generality, allowing representation of more sophisticated, stabilized congestion control schemes.

2. The Node Model

2.1 Node Description

The node, shown in Figure 1, accepts batches of messages from a number of classes, each of which has its own batch size and input stream. Incoming message batches from each stream enter a common hold queue (according to a Poisson arrival process) from where batches are released into the server queue of the message processor, whence they are served on a first-come-first-served (FCFS) basis. Messages are processed one message at a time and the following flow control schemes are operative:

(i) Window control. Each message class is assigned a maximum number of batches allowed in the server and hold queues, its window size;

(ii) Isarithmic control or buffer size limitation. The server queue has an associated threshold value such that no messages are accepted when the queue size reaches this level and a congested mode is entered whereby message acceptance is restricted according to credit availability as discussed next in (iii);

(iii) Pacing control. Once the threshold has been reached messages are accepted only if sufficient credits have been accumulated, until the queue again becomes empty. The number of available credits is set to zero for queue sizes greater than or equal to the threshold value. One credit is issued whenever a message's processing is completed with a resulting queue length less than the threshold value. Batches of messages are accepted from the hold queue, on a FCFS basis, when the number of accumulated credits is greater than or equal to the batch size of the class at the front of the hold queue. On such a message acceptance a number of credits equal to the batch size is consumed, reducing the number of accumulated credits correspondingly. If the server queue size becomes zero, non-congested mode of operation is restored until the threshold is again crossed.

This is basically the node described in [5] with variable message sizes and no time-out

mechanism. The time delay involved in the issue of credits, arising from their own transmission on a network in the physical system, is not represented.

Notation:

Number of message classes:	R
Message processor service rate:	μ messages/unit time
Queueing discipline:	FCFS (both queues)
Class arrival rates:	λ_r batches/unit time ($1 \leq r \leq R$)
Class batch sizes:	b_r messages ($1 \leq r \leq R$)
Class window sizes:	w_r batches ($1 \leq r \leq R$)
Threshold of server queue:	T messages

These parameters may be state dependent in any Markov model, in particular queue length dependent. The direct representation of this system in terms of a Markov process involves a state space with dimensions corresponding to

(i) ordering of batches in the hold and server queues in terms of their classes;
(ii) number of messages in the server queue;
(iii) number of credits accumulated.

The balance equations have no closed form solution because of the global state dependencies of state transition matrix entries on more than one state vector components, c.f. [1], [7]; for example due to blocking. Furthermore, direct solution of the balance equations is infeasible due to the size of the state space. We therefore simplify the formulation of the specification for a model to represent the system so as to reduce the state space size to such an extent that explicit solution of the new balance equations becomes a practicable proposition. The global state dependencies do not vanish so that a closed form solution still cannot be derived.

The following observations may be made:

(i) The system operates in either of two modes:
* congested, when the server queue has not been empty since the last time the threshold was reached;
* uncongested, otherwise.

(ii) Uncongested mode may be represented by some maximum server queue size, C say, where $C \geq T$. Of course it would be meaningless to have $C > T-1+\max(b_r)$ since queue lengths greater than the r.h.s. of this inequality could only occur after an arrival to a queue of length greater than T-1, when no batches are accepted.

(iii) Congested mode may be represented by
* no arrivals for server queue lengths greater than or equal to T.
* arrivals subject to maximum queue length T for server queue length less than T. This is because on a service completion resulting in a queue length below the threshold, a single credit is issued corresponding to a single message completed. Also, when a batch is accepted, the number of credits consumed is equal to the number of messages in the batch. Thus the sum of the number of accumulated credits and the server queue length is constant, equal to T since at the threshold there are zero credits available by definition. This is equivalent to (ii), uncongested mode, with C set to T.

The state space, slightly generalised from this description, is now defined in terms of mode and server queue length; the credit-based control becomes implicit and the model specification greatly simplified.

2.2 The State Space

The state space, S, is the set of ordered pairs defined by

$$S = \{(n,m): 1 \leq n \leq C_m; 1 \leq m \leq M\}$$

where

M is the number of different modes (2 in the previous section);
m is the mode corresponding to state vector (n,m), with m=1 representing congested mode;
C_m is the maximum queue length permitted in mode m, the mode's capacity, so that $C_1=T$;
n is the queue length corresponding to state vector (n,m).

This state space permits a more general flow control scheme than that described in the previous section:

(a) Any number of modes is possible, i.e. $M \geq 2$;

(b) Mode changes may occur when the queue becomes empty (n=0). Initially, n=0 and m>1; $C_m > T$ for a realistic system, otherwise congestion will never occur in the mode m. Subsequently, on entry to a state with n=0 from a state (1,m'), a mode transition to mode m occurs with probability $\pi_{mm'}$. Thus, in the scheme defined in the previous section,

$$\pi_{mm'} = \begin{cases} 1 & \text{if } m=2 \\ 0 & \text{if } m=1 \end{cases}$$

Additional modes may be used to provide stability in the system by introducing intermediate capacities, preventing the immediate onset of repeated congestion. In fact the scheme can be generalised further, providing more stability, via multiple thresholds as discussed in 4.3

(c) A batch from message class r at the front of the hold queue is accepted into the

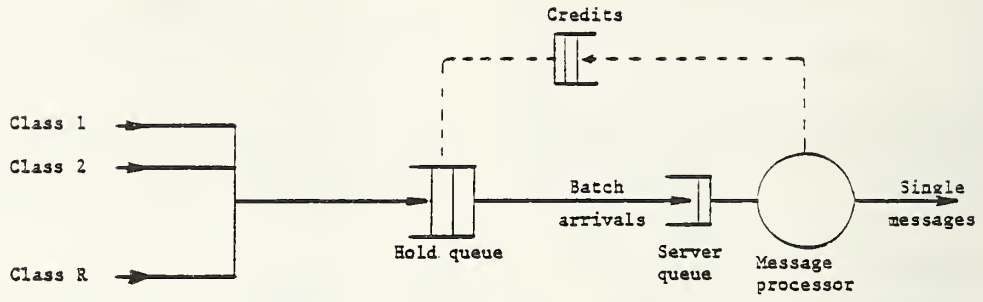


Figure 1. The Flow Controlled Node

server queue in state (n,m) iff the queue capacity would not be exceeded, i.e. iff $n+b_r \leq C_m$. Hence a form of blocking is present even in our approximate model.

(d) On crossing the threshold, congested mode is entered, i.e. a transition to a state (n,m) with $n \geq T$ causes m to be set to 1.

2.3 Implicit Representation of the Congestion Control Mechanisms

The window flow control scheme of the node is represented implicitly in the parameterization of the following two quantities for each class k, $1 \leq k \leq R$:

(a) p_k, the probability that any given message in the hold or server queues belongs to class k. This is chosen to be proportional to the window size as well as the arrival rate of class k. Thus, $p_k \propto \lambda_k w_k$. The value of p_k will also be influenced by the window control through λ_k, as described next.

(b) u_k, the constraining factor on the arrival rate, λ_k, of class k batches, $0 < u_k \leq 1$. These constraints result from the rejection of class k message batches when their number in the hold and server queues is equal to their window size; the effective reducing of λ_k to 0. The constrained arrival rate for class k is then $u_k \lambda_k$.

The value of u_k depends critically on transmission protocol. It is unity if re-transmission is attempted immediately after rejection, or equivalently if rejected batches are stored until they do become acceptable after further processing of sufficiently many class k messages. However, more commonly and inevitably if the total unconstrained message arrival rate (from all classes) exceeds the message processing rate, u_k depends on the distribution of batch classes in the hold and server queues. We denote the constraining factor for class k in state (n,m) by $u_k(n)$: independent of m since the window control is mode-independent.

The approximation given below for $u_k(n)$ is based on the mean value of the length of the combined hold and server queues. We assume that the total number of messages in these queues has the same probability distribution, Z, as the length of the steady state M/M/1 single server queue with

arrival rate $\sum_{k=1}^{R} u_k(n) \lambda_k b_k$,

processing rate μ ,

and capacity $\sum_{k=1}^{R} w_k b_k$,

henceforth denoted by Q. Thus $\{u_k(n)\}$ are defined recursively (and non-linearly) and may be determined iteratively, each being set to 1 initially and successively updated to the required precision. Convergnce is not considered formally here. Our approximation for $u_k(n)$ is derived as follows.

Let $\bar{q}(n)$ be the mean combined length of the hold and server queues in state (n,m). Thus,

$$\bar{q}(n) = \frac{\sum_{j=n}^{Q} j\, Z(j)}{\sum_{j=n}^{Q} Z(j)} \qquad (1)$$

Then the corresponding mean number of batches in the hold and server queues is

$$\bar{m}(n) = \frac{\bar{q}(n)}{\sum_{i=1}^{R} p_i(n) b_i} + 1/2 \qquad (2)$$

where $p_i(n) \propto u_i(n)\, \lambda_i w_i$ and we use the

value of $u_i(n)$ given by the previous iteration to compute $p_i(n)$. The addition of 1/2 allows for the batch currently being processed.

The probability of the queues containing w class k batches, given total length $\bar{m}(n)$, is therefore

$$\binom{\bar{m}(n)}{w}\{p_k(n)\}^w\{1-p_k(n)\}^{\bar{m}(n)-w}$$

for $\bar{m}(n)\geq w$, and we define $u_k(n)$ to be 1 if $\bar{m}(n)\leq w_k$ and

$$\sum_{w=0}^{w_k}\binom{\bar{m}(n)}{w}\{p_k(n)\}^w\{1-p_k(n)\}^{\bar{m}(n)-w} \quad \text{if } \bar{m}(n)\geq w_k \tag{3}$$

Notes:

(i) In general, $\bar{m}(n)$ is not an integer, and the generalized factorial function is used in the combinatorials to provide an interpolated estimate for $u_k(n)$ as a function of mean queue length.

(ii) This approximation assumes primarily that the hold queue length is always the same for any given server queue length. A more precise analysis which does not require this assumption is given in [3].

The credit allocation control scheme is reflected implicitly in the effective batch arrival rate in state (n,m),

$$\Lambda_{nm} = \frac{\sum_{\substack{i=1 \\ n+b_i\leq C_m}}^{R} u_i(n)\lambda_i b_i}{\sum_{\substack{i=1 \\ n+b_i\leq C_m}}^{R} p_i(n) b_i} \tag{4}$$

so that $\sum_{n+b_i\leq C_m} u_i(n)\lambda_i b_i$

= overall message arrival rate in state (n,m)

= (effective batch arrival rate) * (mean accepted batch size)

as required. Note that for degenerate b_i's or w_i's,

$$\Lambda_{nm} = \frac{\sum_{n+b_i\leq C_m} u_i(n)\lambda_i}{\sum_{n+b_i\leq C_m} p_i(n)} \tag{5}$$

again as required.

Note too that if batch arrival rates are unconstrained by window control, i.e. if $u_k(n)=1$ for each message class k and queue length n, Λ_{nm} may be inaccurate since, for example, one arrival stream may dominate through its high arrival rate and yet have a small, highly constraining window size.

2.4 The Balance Equations

2.4.1 Further Definitions

Before the balance equations can be derived, some more definitions are required:

$B_{nm}(k)$ = probability that the system is blocked in state (n,m) waiting to accept a batch of class k from the front of the hold queue, $1\leq k\leq R$

B_{nm} = probability of any blocking in state (n,m)

$$= \sum_{k=1}^{R} B_{nm}(k) \tag{6}$$

We make the following (intuitively sound) approximation:

$$B_{nm}(k) = \begin{cases} A_{nm}p_k(n) & \text{if } b_k > C_m - n \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

where A_{nm} is the probability that there is at least one arrival of any class outstanding in the hold queue in state (n,m). The parameter A_{nm} is important in that its choice of value allows various different message batch acceptance protocols to be represented. Some examples of relevant value assignments are:

(i) For a re-transmit batch acceptance protocol in which a message batch is discarded if it cannot be accepted into the server queue immediately, no blocking occurs and $A_{nm} = 0$ for all $(n,m) \in S$.

Such a protocol is good as regards buffer protection and no hold queue is required. Throughput is not reduced significantly and it is the easiest case to model; the credit control scheme is redundant. However, the protocol is unfair to large batches which do not have bounded waiting time for service and suffer severely if there is heavy small batch traffic.

(ii) Hold queue with FCFS discipline and unbounded buffer size (unlimited queue length). For state $(n,m)\in S$,

let $\Lambda(n) = \sum_{k=1}^{R} b_k\lambda_k u_k(n)$ (8)

the total, window-constrained, external

message arrival rate in state (n,m).

If $\Lambda(n) \geq \mu$, the system is overloaded and there will always be a blocked arrival; $A_{nm} = 1$.

If $\Lambda(n) < \mu$,

A_{nm} = Prob(total no. of messages in hold & server queues > n : server queue length = n)

Considering the whole system as an M/M/1 single server queue, this yields

$$A_{nm} = \text{Prob}(\text{queue length} > n : \text{queue length} \geq n)$$

$$= 1 - \frac{Z(n)}{\sum_{j=n}^{Q} Z(j)} \qquad (9)$$

in the notation of 2.3. This is the parameterization we choose for our model. Clearly, further approximation has been introduced, but note that the result is exact for $\Lambda(n)=0$ and as $\Lambda(n) \to \infty$ with maximum allowed queue size approaching infinity. Furthermore, one would expect the inaccuracy introduced to be considerably less than that arising from the other approximations in the model.

With the problem now fully defined it is a relatively simple matter to write down the balance equations for the scheme. These may be found in [3], together with a discussion of some numerical aspects of their solution. We note here that the size of the state space is $1+(M-1)T+\max_{1 \leq m \leq M} C_m$, and the number of elements in the transition matrix is the square of this value. Typically, M will be small, e.g. 2 in our original specification, and T and $\{C_m : 1 \leq m \leq M\}$ not large with respect to computational feasibility; in any case, the state space size is only linear in T.

No storage optimization techniques have been found necessary, but the transition matrix is clearly suitable for sparse matrix processing techniques. These could be enhanced further using certain properties of the model. For example by considering separately the mode changes which can occur only in very few states; the intervening states enterable may then be considered to have only one component, the queue length. In particular, the state (n,1) with n>T can transit only to the state (n-1,1) and thence, eventually, to (T,1). Thus all super-threshold states could be lumped together into a single composite state. This would produce a more efficient implementation with respect to both execution time and storage, but results in loss of potentially important detail, viz. the queue length distribution above the threshold.

3. Validation

3.1 The Simulation Model

The analytic model defined in the previous sections was validated by comparing its predictions with those of a simulation model which represents the node's operating characteristics explicitly (although very much less efficiently). Whilst it is conceded that ultimate validation must be based on data monitored on at least one existing physical node, agreement with simulated results certainly supports confidence in our model; its implementation, but most importantly its approximations.

The simulation model is an algorithmic representation of the node defined in 2.1. It models explicitly the flow of messages in each class and the window flow control mechanism as well as the credit allocation/threshold congestion control scheme which is equivalent to two modes of congestion. All interarrival times and the message processor service times are drawn from negative exponential probability distributions (consistent with our Markov model), and messages rejected due to window flow constraints are discarded: a stream's arrival rate is effectively shut down to zero when it attempts to violate its window size limit.

The durations of the time periods simulated were chosen to be long enough to yield sufficiently small standard error estimates on the derived performance measures: of the order of 5%, see 3.3. Both the analytic formulae and simulation are simply programmed in APL, and run on an IBM 5100, the efficiency of which is conducive to performance evaluation by wrist watch! The analytic results for each test case required around 5 minutes of CPU time, whereas the corresponding simulation runs lasted some 10 hours.

3.2 Node Test Cases

In any test, the total message arrival rate from all classes must be at least close to the message processor's service rate in order that any congestion control become operative. Relatively low arrival rates were used to validate the implementations of the models - giving results close to those of corresponding M/M/1 queues, as required.

Similarly, for a node with all batches consisting of only a single message, the maximum queue length in all modes is the threshold value, since congested mode will be entered as soon as the queue reaches this size: the modes are degenerate. Here, the node is precisely a regular single server queue (M/M/1 under our Markovian

assumptions) with the super-threshold states aggregated into one. Thus, for a single arrival stream of unit-sized batches with window size Q, in the notation of 2.3, we should have

$$P_i = Z(i) \quad \text{for } 0 \le i < T$$
$$P_T = \sum_{j=T}^{Q} Z(j) \qquad (10)$$

where we drop the redundant mode subscript. Now, batches can be blocked in the hold queue only in state (T,1) and so the balance equations for our analytic model reduce (via the conventional method for the M/M/1 analysis) to

$$P_{i+1} = \rho(i)P_i \quad \text{for } 0 \le i < T-1$$
$$P_T = (1-A_T)^{-1}\rho(T-1)P_{T-1} \qquad (11)$$

where $\rho(i) = \lambda(i)/\mu(i+1)$ for message arrival rate $\lambda(i)$ and processing rate $\mu(i)$ in state (i,1), and A_T is the probability of there being a batch waiting in the hold queue in state (T,1), see 2.4. Now, $Z(i+1)=\rho(i)Z(i)$ for $0 \le i < Q$, so we require

$$\{1-A_T\}^{-1}Z(T) = \sum_{j=T}^{Q} Z(j) \qquad (12)$$

which is satisfied by our definition of A in 2.4.

Numerical results are in agreement, but this merely provides some validation of the implementations of the analytic and simulation models. Note that although the modes are degenerate, for finite window sizes the arrival streams are not; they result in a multi-class single server queue.

The following selection of nodes defined for our validation was chosen to demonstrate the most significant characteristics of the flow controls:

(1) A node with 2 arrival streams and the threshold value sufficiently high for both the credit allocation and window flow control schemes to have significant effects. Note that a threshold above the sum of the products of the window size and batch size for each class would never be reached.

(2) The same node with the threshold reduced so that the credit allocation control scheme becomes dominant.

(3) The node with total (unconstrained) message arrival rate much higher than the message processor's service rate - a saturated node.

(4) A node with a third arrival stream of single message batches with relatively high arrival rate and window size. Such a stream allows the server queue length to remain at its threshold value after a service completion (in congested mode, of course), due to acceptance from the hold queue of a message batch of unit size.

The numerical parameterizations of these specifications are given in Tables 1-4, along with their performance predictions. These test cases are a sub-set of a much more comprehensive set, revealing our most significant results as discussed in later sections.

3.3 Estimation of Performance Measures

The primary performance measure, computed by both the analytic and simulation models for each test case (3.2), is the probability distribution of the server queue length. Secondary quantities then follow immediately, for example throughput, mean and standard deviation of queue length, expected message waiting time (through Little's Law). In some simulation runs, we also estimated the proportion of time that:

(a) Each arrival stream is inactive due to the window contol. In this way the validity of our approximation for $\{u_k(n)\}$, 2.3, could be judged directly.

(b) The hold queue is non-empty, with a class k batch at the front, $1 \le k \le R$. Hence the accuracy of our assignment of values to $\{A_{nm}\}$, 2.4, can be assessed for cases more complex than that discussed in 3.2.

(c) The node is in each mode. This can provide further validation by comparison with the appropriate marginal state space probabilities of the analytic model.

In each simulation run, initially both the hold and server queues are empty and the node is in non-congested mode. No data is collected until a period of 5 time constants has elapsed. The value used for the time constant is the reciprocal of the mean state transition rate, averaged over all states in the analytic model. We used primarily the sampling or "snapshot" method of data collection, described below, for our statistical analysis. Cumulative techniques were also used to provide an alternative approach, but the resulting estimates are not significantly different from, in fact are almost identical to, those based on the snapshot data. The cumulative variant is described in [4].

Snapshots of the node's state were sampled at a set of time points uniformly spaced at intervals of 2 time constants, the first at time equal to 5 time constants. If server queue length i is observed n_i times in a sample of N snapshots, the estimate for the equilibrium probability of queue length i is $\hat{p}_i = n_i/N$, $0 \le i \le I$, where I is the maximum possible queue length. Assuming that the sample is independently identically distributed according to true queue length probabilities $\{p_i: 0 \le i \le I\}$, the random variable n_i has binomial distribution with mean Np_i and variance $Np_i(1-p_i)$.

Thus, $\hat{p}_i$ has expected value p_i (as

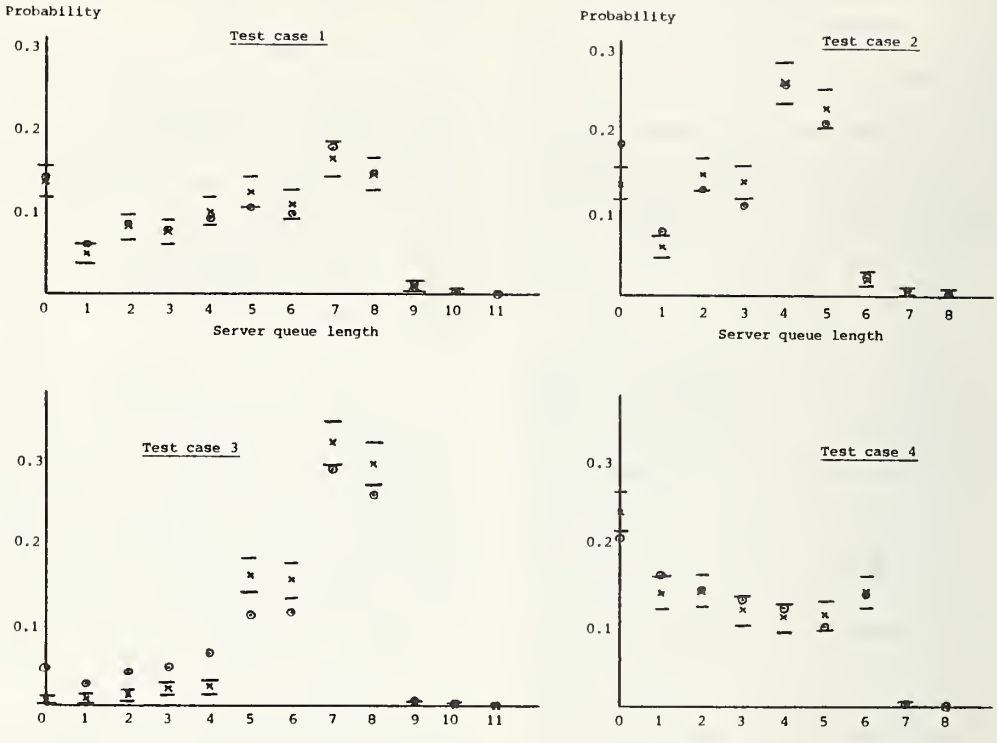


Figure 2. Predicted Server Queue Length Probability Distributions

required) and variance $p_i(1-p_i)/N$. Hence, $\hat{p}_i$ has standard error estimate

$$\sigma[\hat{p}_i] = \{\hat{p}_i(1-\hat{p}_i)/N\}^{0.5} \qquad (13)$$

Since N is proportional to the (simulated) duration of the run, D say, the standard errors are proportional to $D^{-0.5}$. Thus to reduce the standard error estimates by a factor of f, the simulation run time must increase by a factor of f^2.

The mean queue length is estimated as

$$\sum_{i=1}^{I} i\hat{p}_i$$

and so has standard errror

$$\{\sum_{i=1}^{I} (i\ \sigma[\hat{p}_i])^2\}^{0.5} \qquad (14)$$

A numerical comparison of these performance measures as computed by the analytic and simulation models is presented in Tables 1-4, with the goodness-of-fit assessed by the chi-squared test. Histograms derived for the server queue length probability distributions are shown in Figure 2.

3.4 Analysis and Interpretation of the Numerical Results

3.4.1 Overview and General Observations

From tables 1-4 and the plots in Figure 2 it can be seen that the analytic and simulation results exhibit the same general characteristics: in terms of mean queue length, the distribution itself and their standard errors. The distribution is essentially bimodal (with the exception of the case with high arrival rate discussed below), decreasing from the idle probability (queue length

zero) to a trough before the threshold queue length, increasing to a maximum at or immediately below the threshold and falling away sharply for queue lengths above the threshold. The corresponding single server queue results, with window flow controlled arrival rate and all super-threshold states lumped into one, bear little resemblance.

Apart from the general bimodal shape, there are other more local variations in the distributions. Denoting the probability of queue length n by P(n), for test cases 1-3, $P(0)>P(1)<P(2)>P(3)<P(4)$, whereas in case 4 the probabilities are monotonically decreasing for sub-threshold queue lengths. This is simply explained by the absence of a single message batch stream in all cases other than the fourth. The (queue length) transition 0->1 is then invalid and queue length 1 can only result from a transition 2->1, on a message completion. Thus $P(1)<P(2)$ and the effect propagates to P(3) for precisely the same reason. Similarly in cases 1-3, $P(T-1)>P(T)$ for threshold value T. This follows since the transition T->T is invalid (acceptance of a blocked single message batch on service completion) and so all super-threshold states must transit monotonely to state (T-1,1). Propagation analogous to that discussed above results in $P(T-3)>P(T-2)<P(T-1)$ in cases 1 and 2, the effect being less pronounced in case 3 because of the counteracting high arrival rate.

In contrast, note that in case 4, the presence of the unit sized batches allows the transition T->T, yielding very emphatically $P(T)>P(T-1)$. Further, $P(T-1)<P(T-2)$, explained by the existence of (T-1)->T (blocked arrival) transitions: no similar transition is possible from queue length T-2 and the effect is outweighed by the others in cases 1-3.

3.4.2 Goodness-of-fit

It is clear from our tables and graphs that there is very variable agreement between the analytic and simulation results: from very good (test case 1) to poor (case 3). The mean queue lengths for cases 1,2,4 certainly do not differ significantly (in terms of their standard error estimates); even case 3 is not outrageous here. However, more rigorous quantitative testing must be based on whole distributions. The proportion of analytically computed probability points lying outside the 2 x standard error confidence bands on the simulation results (Figure 2) are 0, 37, 100 and 12 per cent for cases 1-4 respectively. Excluding case 3, only one point differs by more than 3 standard errors (queue length 0 in case 2). Case 3 is the saturated node and is discussed in some detail in the next section. Note here that although quantitatively poor, its analytic predictions still exhibit general characteristics not inconsistent with the simulation.

Our primary statistical test for goodness-of-fit is the chi-squared test. This again reflects the variability of agreement, with values of the test statistic in the range 12-200 on 8-11 degrees of freedom (Tables 1-4). The corresponding percentage points indicate a fit which is good for case 1 (about 75%), reasonable for case 4 (96%) and poor for case 2 (99.9%); effectively 100% for case 3. These results are not too impressive in themselves, indeed rather surprising in case 2. But the chi-squared test is well known for its extreme and unforgiving sensitivity: just one or two points differing significantly yield a large contribution in the test statistic which consequently shows a poor fit regardless of the sizes of the contributions from the other points. This is precisely the situation in test cases 2 and 4, as well as many others not reported here, with particularly large contributions from small queue lengths (see the last columns in the tables). If such points are corrected or omitted, a good fit is indicated. Thus, although the chi-squared test may suggest a model worthless, it may be that for a (majority) sub-set of queue lengths the model provides a good representation - actually supported by this self-same test which may reject a whole theory on the basis of one point.

Our model, then, appears to provide a good, statistically supported representation of queue length behaviour if underload situations are excluded, This is especially true near the threshold, precisely where the control schemes modelled are most crucial and prediction is most important. Some possible causes for the inadequacies identified in the model are revealed next.

3.4.3 Interpretations

(1) For many of the node specifications examined, cases 1-3 here, the simulation predicted a sharper peak in the queue length distribution at (or just below) the threshold. This is quite consistent with its lower predictions for the probability of zero queue length in that non congested mode would be entered less frequently. Thus the queue capacity would be greater than the threshold value less often, and hence the threshold will provide a more dominating constraint. In more general terms also, some smoothing of the peak in the analytic model is consistent with the aggregate type of representation of window control: discussed further below.

(2) As expected, the credit allocation control (CAC) dominated cases give the best agreement, e.g. case 1: CAC is modelled explicitly and in greater detail than the window flow control (WFC), represented only in terms of aggregate effects on

arrival rates. Moreover the WFC has representation independent of the CAC and so of the threshold value. Thus, if it is inadequate, the analytic model cannot be expected to be accurate since its own input (arrival rate) parameters would be incorrect. This applies regardless of the threshold value, so that by a "CAC dominated" test case we really mean one in which WFC has little effect on arrival rates and not necessarily merely one with low threshold.

The good agreement obtained in case 1 is consistent with this argument, but greater positive support is provided by the poor results of test case 3 in which batch arrival rates were high and consequently adjusted by the WFC to a considerable extent. Certainly the significant probabilities predicted for queue lengths 0 and 1 are counter-intuitive, and we postulate that the inaccuracy arises from our representation of WFC. Exploring this possibility further, we tested a case with two arrival streams, one with much greater arrival rate but small window size. The resulting adjustments to the batch arrival rates were therefore substantial, even though the node was not overloaded. This test gave a chi-squared statistic of around 500 on 9 degrees of freedom, strongly suggesting that our arrival rate adjustment formula (2.3) is inadequate.

On re-computing the analytic results for this node specification and test case 3, using the WFC-constrained arrival rates estimated by monitoring the corresponding simulations (3.3 (b)) in place of the adjustment formula, much better agreement was obtained. A need for improvement in our representation of WFC is clearly identified, and may be provided to some extent through a refinement of our aggregate approach, [3]. A more detailed representation would be preferable, but recall that it is impractical for this to be fully explicit (2.1). Note that another similar possible source of inaccuracy is the formulation of the batch class selection probabilities, $\{p_k(n)\}$ of 2.3.

(3) Finally, and in the same vein as (2), the parameterisation of the blocking probabilities through $\{A_{nm}\}$ may lead to imprecision. Again the WFC representation is also significant through $\{p_k(n)\}$, as well as $\{A_{nm}\}$ to a lesser extent. However, tests using values for the blocking probabilities estimated in corresponding simulation runs (compare (2)) were inconclusive.

4. Applications and Enhancements of the Model

4.1 Performance Metrics

Most of the required performance measures can be derived from the state space probabilities, $\underline{P}$. These are as follows:

(i) Server queue length probability distribution, marginal probabilities of the joint probability distribution, $\underline{P}$;
(ii) Server utilization, following directly from (i);
(iii) Server throughput;
(iv) Mean and standard deviation of the server queue length;
(v) Distribution of the time delay for a message batch to pass through the message processor, a simple weighted convolution;
(vi) The power of the node, defined as

$$\frac{\text{Throughput of node}}{\text{Mean message time delay in both queues}} = \frac{(\text{Throughput})^2}{\text{Sum of mean queue lengths}}$$

by Little's Law. Now the throughput may be derived from the node's state space probabilities and the sum of the mean queue lengths obtained by considering the whole system as an M/M/1 single server queue with window flow constrained arrival rate (2.3).

4.2 Experimentation

Having derived expressions or algorithms for the performance measures listed in the previous section, one can attempt to optimize the node's performance in various ways, by selection of appropriate parameter values. For example,

(i) In the graphs of throughput and power vs. mean time delay or mean queue length (related by Little's Law) for messages in the system, one can identify a knee and a peak respectively. Optimization would involve choosing model parameters to achieve positioning of these points which is best according to some objective; e.g. maximum throughput subject to some upper bound on mean time delay (reflecting provision of some pre-determined minimal service quality).
(ii) The distribution of the server queue length is important, particularly if service rate depends on queue size. Clearly a low probability of zero queue length is required (minimal idle time). In addition, the "spread" of the distribution should not be too great, otherwise the variance of the number of messages in the queue will be high, reflecting a highly variable message delay time and possible instability; a most undesirable effect, psychologically at least. Thus the ideal shape for the queue length distribution is that of a sharp peak at a length greater than zero. Furthermore, for a queue length dependent service rate, one would like to be able to choose parameter values such that the peak was located near the queue length yielding maximum service rate (the queue is finite, recall); in this way, throughput should be optimized also.

Considerable experimentation with parameter values, such as the threshold, mode capacities, batch and window sizes, is possible, since the significant characteristics of the flow control scheme are parameterized explicitly. Particular examples are suggested in [3].

4.3 Extensions to the Model

An immediate extension is to incorporate state-dependent parameters into the model; arrival and service rates in particular. Much more sophisticated optimization may then be undertaken, as in 4.2. Dynamic flow control schemes are being used increasingly, particularly in networks with distributed control functions. The parameters of such schemes are adjustable according to the current state of the network. For example, as congestion increases at any particular node, the node can reduce, or even shut down to zero in severe cases, the window sizes of some or all message classes. Of course the physical mechanism would involve transmission of control information on the network, requiring much more complex modelling which would be impracticable in our case. However it is a simple matter to incorporate queue length or mode dependent window sizes, w_r, represented by p_r and u_r in the balance equations. In this way, the distribution of messages with respect to class, entering the message processor queue may be controlled according to the current state.

State dependent mode capacities may be used to model stabilizing mechanisms which may operate physically by issuing variable (fractional or multiple) credits. Indeed in general, mode capacity is equivalent to the size of credits issued on sub-threshold message completion. Although a less trivial change, the balance equations are easily modified to accomodate such dynamic mode capacities. A form of dynamic message acceptance protocol is represented in this way.

A novel stability mechanism may be provided by the use of multiple thresholds. In our model, there is only one threshold, the only significance of which is that the mode, or queue capacity, may change when it is reached from below. The mode always becomes 1, congested, but there is no analytical reason for this restriction or why a mode transition probability matrix similar to $\underline{\underline{\pi}}$ could not be used. Furthermore, such transitions are quite realistic and feasible, and capable of representation in the balance equations, on approaching the threshold from above as well as below; one could view the existing model as having unit mode transition matrix on reaching the threshold from above.

Using the possibility of a mode change at a queue length of n as the defining criterion for a threshold, the empty queue is also a threshold. Extending this argument, any number of thresholds may be defined, the mode being subject to change at any; approaching from above or below. In this way, stabilizing mechanisms may be represented via entry to intermediate modes or use of a threshold band to delay revertion to the old mode on returning through a threshold. The threshold band operates as follows:

(i) The band consists of two values for the queue length, T_A and T_B with $T_A < T_B$ for thresholds A and B respectively.

(ii) On reaching threshold B from below, the mode may change; to a "more congested" mode. For example, m may change from 2 to 1 in our test case with M=2 at the threshold with value T. On approach from above, no mode change occurs.

(iii) Similarly, on reaching A from above, the mode may change; to a "less congested" mode. In the same example, m may change from 1 to 2 when the queue becomes empty. On approach from below, no mode change occurs.

It can be seen that this mechanism will prevent instability caused by oscillation of the queue length about a single threshold value having inverse mode transitions for approach from opposite directions. In fact in our example we have precisely a threshold band, with $T_A = 0$ and $T_B = T$, but far greater generality is possible.

5. Conclusion

An analytic model has been developed which can represent a wide range of nodes or sub-systems in queueing networks with congestion control schemes. The equivalence of the credit- and mode-based control schemes, together with certain approximations yield a state space sufficiently small to permit feasible, direct, numerical solution of its balance equations; even for complex nodes with large buffer sizes. At the same time, enough detail of a modelled system, in terms of its parameters, is retained to achieve great generality; mainly through the acceptability of unrestricted state dependence of the parameter values.

The type of control modelled is node-to-node rather than end-to-end, based on single buffer capacity constraints and restrictions on the arrival processes to one processor. The control is localized, reflecting message pacing and single node protection, and is eminently suitable for application to networks with distributed flow control. End-to-end flow control is modelled via the window size parameters, although its explicit implementation is not.

Validation to date is encouraging, suggesting that our model provides a good analytic representation of nodes' behaviour, for server queue lengths near the threshold in particular.

The choices of threshold value and mode capacities are crucial for optimizing performance, and this domain is the most important for prediction purposes. We therefore expect that the model can be a suitable tool for performance prediction in the next stage of our research - experimentation with multiple modes and thresholds in the development of dynamic stability mechanisms, see 4.3.

Perhaps of less obvious importance, it should be pointed out that in addition to providing a rich class of models, this modelling process has increased insight into and understanding of the corresponding physical processors; resulting in the proposals for new or extended congestion control schemes. A network consisting of nodes all of which possess flow control mechanisms of these types could provide dynamically well paced, efficient, fair and secure communication between end users.

The research reported in this paper originated during my recent visit to IBM Watson Research Center, Yorktown Heights, New York. I would like to extend my thanks to Frank Moss, Mischa Schwartz, Paul Green, but above all to Parviz Kermani, for all their help and advice.

References

[1] Chandy, K.M., Howard, J.H., Towsley, D.F., "Product Form and Local Balance in Queueing Networks", J.ACM 24,2.

[2] Davies, D., "The Control of Congestion in Packet Switching Networks", IEEE Transactions on Communications, Vol. COM-20, No. 3, June 1972.

[3] Harrison, P.G., "Analytic model of a Communication Network Node with Flow Controls", IBM Research Report RC8832, Research report DoC81/9, May 1981, Dept.of Computing, Imperial College, London SW7 2BZ, England. Also submitted to Computer Networks.

[4] Harrison, P.G., "Performance Prediction of a Flow Control System using an Analytic Model", Research report DoC81/10, June 1981, Department of Computing, Imperial College, London SW7 2BZ, England.

[5] Kermani, P., Bharath-Kumar, K., "A Congestion Control Scheme for Window Flow Controlled Computer Networks", IBM Research Report No. RC8401.

[6] Kleinrock, L., Gerla, M., "Flow Control: A Comparative Survey", IEEE Transactions on Communications, Vol. COM-28, No. 4, April 1980.

[7] Lam, S.S., "Queueing Networks with Population Size Constraints", IBM J. Res. Develop., July 1977.

[8] Lam, S.S., Reiser, M., "Congestion Control in Store and Forward Networks by Buffer Input Limits", IEEE Transactions on Communications, Vol. COM-27, No. 1, January 1979.

[9] Wecker, S., "DNA: The Digital Network Architecture", IEEE Transactions on Communications, Vol. COM-28, No. 4, April 1980.

Table 1. First Node Test Case

MESSAGE CLASSES:	1	2
ARRIVAL RATES	5.000	3.000
BATCH SIZES	2	4
WINDOW SIZES	3	2

MESSAGE PROCESSING RATE = 20.000
THRESHOLD VALUE = 8
NON-CONGESTED MODE CAPACITY = 11

QUEUE LENGTH	PROBABILITY: ANALYTIC	PROBABILITY: SIMULATION	STANDARD ERROR	PROBABILITY: M/M/1 QUEUE	CHI SQUARED CONTRIBUTION
0	.142	.137	.010	.068	.224
1	.060	.048	.006	.074	3.287
2	.086	.082	.008	.080	.239
3	.078	.075	.007	.086	.152
4	.092	.101	.009	.090	1.092
5	.106	.124	.009	.092	4.147
6	.099	.110	.009	.091	1.535
7	.178	.165	.010	.087	1.174
8	.147	.146	.010	.331	.009
9	.008	.010	.003	.000	.332
10	.003	.003	.001	.000	.098
11	.001	.000	.000	.000	1.222

SAMPLE SIZE = 1250
LENGTH OF TIME PERIOD SIMULATED = 112.095

MEAN QUEUE LENGTH
ANALYTIC: 4.497
SIMULATED: 4.562
STD ERROR: .140

CHI-SQUARED STATISTIC = 13.510 WITH 11 DEGREES OF FREEDOM

Table 2. Second Node Test Case

MESSAGE CLASSES:	1	2
ARRIVAL RATES	5.000	3.000
BATCH SIZES	2	4
WINDOW SIZES	3	2

MESSAGE PROCESSING RATE = 20.000
THRESHOLD VALUE = 5
NON-CONGESTED MODE CAPACITY = 8

QUEUE LENGTH	PROBABILITY: ANALYTIC	PROBABILITY: SIMULATION	STANDARD ERROR	PROBABILITY: M/M/1 QUEUE	CHI SQUARED CONTRIBUTION
0	.184	.136	.010	.068	15.122
1	.078	.059	.007	.074	5.592
2	.129	.148	.010	.080	3.516
3	.110	.138	.010	.086	9.025
4	.257	.260	.012	.090	.054
5	.211	.228	.012	.602	1.790
6	.022	.021	.004	.000	.076
7	.007	.006	.002	.000	.281
8	.003	.004	.002	.000	.022

SAMPLE SIZE = 1250
LENGTH OF TIME PERIOD SIMULATED = 126.505

MEAN QUEUE LENGTH
ANALYTIC: 2.954
SIMULATED: 3.144
STD ERROR: .091

CHI-SQUARED STATISTIC = 35.477 WITH 8 DEGREES OF FREEDOM

Table 3. Third Node Test Case

MESSAGE CLASSES:	1	2
ARRIVAL RATES	10.000	6.000
BATCH SIZES	2	4
WINDOW SIZES	3	2

MESSAGE PROCESSING RATE = 20.000
THRESHOLD VALUE = 8
NON-CONGESTED MODE CAPACITY = 11

QUEUE LENGTH	PROBABILITY: ANALYTIC	PROBABILITY: SIMULATION	STANDARD ERROR	PROBABILITY: M/M/1 QUEUE	CHI SQUARED CONTRIBUTION
0	.046	.006	.002	.001	42.705
1	.026	.008	.003	.002	15.091
2	.041	.013	.003	.003	24.282
3	.047	.021	.004	.005	17.867
4	.064	.023	.004	.007	33.280
5	.110	.159	.010	.010	27.326
6	.113	.153	.010	.016	17.901
7	.287	.320	.013	.024	4.589
8	.256	.294	.013	.932	7.401
9	.006	.002	.001	.000	4.576
10	.003	.001	.001	.000	1.680
11	.001	.000	.000	.000	1.068

SAMPLE SIZE = 1250
LENGTH OF TIME PERIOD SIMULATED = 104.202

MEAN QUEUE LENGTH
ANALYTIC: 5.881
SIMULATED: 6.516
STD ERROR: .162

CHI-SQUARED STATISTIC = 197.765 WITH 11 DEGREES OF FREEDOM

Table 4. Fourth Node Test Case

MESSAGE CLASSES:	1	2	3
ARRIVAL RATES	8.000	2.000	2.000
BATCH SIZES	1	2	3
WINDOW SIZES	6	2	1

MESSAGE PROCESSING RATE = 20.000
THRESHOLD VALUE = 6
NON-CONGESTED MODE CAPACITY = 8

QUEUE LENGTH	PROBABILITY: ANALYTIC	PROBABILITY: SIMULATION	STANDARD ERROR	PROBABILITY: M/M/1 QUEUE	CHI SQUARED CONTRIBUTION
0	.204	.237	.012	.163	6.567
1	.160	.139	.010	.146	3.500
2	.143	.142	.010	.131	.013
3	.131	.118	.009	.117	1.782
4	.121	.109	.009	.105	1.628
5	.098	.112	.009	.094	2.319
6	.138	.141	.010	.244	.098
7	.003	.003	.002	.000	.024
8	.001	.000	.000	.000	.868

SAMPLE SIZE = 1250
LENGTH OF TIME PERIOD SIMULATED = 88.029

MEAN QUEUE LENGTH
ANALYTIC: 2.672
SIMULATED: 2.637
STD ERROR: .090

CHI-SQUARED STATISTIC = 16.798 WITH 8 DEGREES OF FREEDOM

Case Studies

RESOURCE MANAGEMENT IN A DISTRIBUTED PROCESSING ENVIRONMENT --COMPUTER RESOURCE SELECTION GUIDELINES

Eva T. Jun

Resource Planning and Measurement Branch
Division of Operations
Office of Computer Services and
Telecommunications Management
Office of the Assistant Secretary for
Management and Administration
U.S. Department of Energy
Germantown, MD 20874

In an installation where computer resources of various types and sizes are either available or feasible, resource management and planning personnel have the responsibility for ensuring that user requirements are met in the most cost-effective manner. This paper is a tutorial on what considerations should be a part of the computer resource selection process for any given application.

Key words: Computer resource selection; cost; long range plan; system requirements; user requirements.

1. Introduction

The purpose of these guidelines is to ensure that the selection method used for automatic data processing (ADP) equipment and services will lead to the acquisition of an appropriate and cost-effective ADP resource for a given ADP application system.

These guidelines outline the process for selecting the resource upon which an application is to be run once user requirements have been defined.

With the emergence of distributed processing technology, resource management and planning personnel have a wider choice of equipment selection. In an installation where computer resources of various types and sizes are either available or feasible, planners have the task of matching a resource to each application.

2. ADP Resource Selection Process

2.1 Timing

The timing of ADP resource selection is critical to both the user and to the data processing department. For new applications, ADP resource selection should take place during the initiation phase of system development. As illustrated in Figure 1, resource requirements should be identified and quantified after user requirements are known and either before or concurrently with preliminary design. For existing systems undergoing expansion (running on dedicated processors), ADP resource selection should take place at least 6 months before projected saturation (much longer in certain Federal sectors). However, for existing systems undergoing major modifications, the appropriateness of the original resource selected should also be re-evaluated.

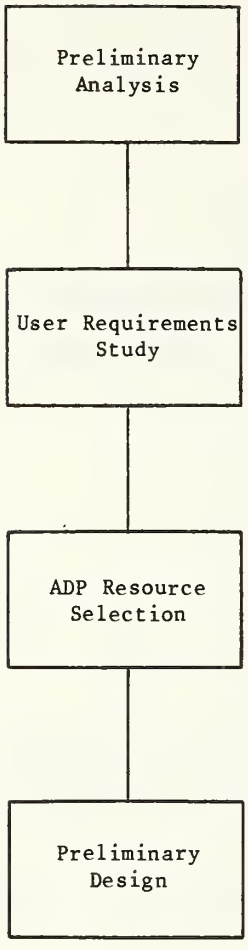


Figure 1. Application System Initiation Phase

2.2 Understanding User Requirements

A distinction must be made between user requirements and system requirements. User requirements should accurately reflect the functional needs of the user stated in terms that the user understands, whereas system requirements should define the computing environment. Quantities in user requirements should be stated in the units the user utilizes to measure his tasks, not in terms of computer system terminology. Figure 2 illustrates the kinds of critical user requirements that must be fully understood.

- Security/privacy
- Nature of interface with other systems:
  - One-way or two-way data exchange
  - Frequency of interfacing
  - Data volume
  - Response time requirement
- Nature of processing:
  - Computation or data management extensive
  - Accuracy requirement
- Online storage requirements in terms of:
  - Number of files per system
  - Average number of records per file
  - Average record length
- Geographical distribution of users
- Peak load conditions:
  - Maximum number of simultaneous users or processes
  - Peak frequency of transaction
  - Turnaround time requirement
- Input/output requirement:
  - Input - screen formatting, data editing, and validation
  - Output - format and volume
- Frequency of usage:
  - Average number of interactive and batch sessions per day
  - Average length of connect time per session

Figure 2. User Requirements

2.3 Long Range Plan

Although the processing requirements of most applications can be satisfied by most general purpose computers, conformity to the data processing department's long range plans is an important factor in the selection of a computing resource. The following paragraphs detail the major points of one such plan, that used at the Department of Energy (DOE) Headquarters Administrative Computer Center in Germantown, Maryland.

1. The Headquarters large scale host computer will be used for systems that require:

   - Online integration of multiple data bases--with current technology, this can best be accomplished on the host computer because of response time constraints;
   - A large volume of data, because of the economy of scale offered in mass storage configurations on large scale processors;
   - A large amount of "number crunching," where the turnaround time would be intolerable on smaller scale processors.

2. Classified material will be processed on dedicated classified processors.

3. Satellite processors are small to medium scale processors used to offload work from the host processors. They are appropriate for systems that do not have the specific requirements outlined above for a host computer. Satellite processors can be installed in field offices as well as Headquarters for the convenience of users. Factors to be considered in the assignment of users to any satellite processor will be based on functional requirements; geographic location; user organization; and the need for access to data, local control, and the relative capacity of the processors. Evaluation of these factors must be made on a case-by-case basis.

4. Office system processors may be purchased and used by specific organizations within user departments where proven to be cost-effective. Mini/micro systems are appropriate for office automation systems because they offer user control, fast turnaround for printer output, and (potential) economy.

   Data processing may be performed on user-owned processors when the sustained processing load of parochial data justifies the acquisition of a computer for data processing and when capacity of other suitable processors is not available.

5. The use of computing resources through interagency resource sharing (for Federal Government agencies) or commercial timesharing services must be considered when the following conditions are present:

   - When needed as back up for in-house facilities when in-house computing facilities are temporarily incapacitated or when there is a need for a temporary means of providing service due to the saturation of in-house computing facilities and urgency of user needs;
   - When it is not feasible to provide proprietary software packages on the in-house computers;
   - When the use of special hardware capability is necessary;
   - When there is a need to access proprietary data bases.

   Interagency resource sharing is available from several Federal data processing centers that serve Federal agencies throughout the country. The General Services Administration (GSA) assists agencies in screening available Government ADP resources. Federal Property Management Regulations (FPMR) require that interagency ADP resource sharing be considered prior to acquiring ADP resources from commercial sources.

2.4 System Requirements

User requirements must be translated into system requirements in order to determine the computing environment necessary to accomplish the user's tasks. The size of the application system plays a major role in establishing system requirements. Once application size has been determined, the physical resources needed to support it can also be determined, thereby allowing selection of a processor with sufficient capacity (provided other requirement areas can be satisfied).

Determining the size of an application system before it has been developed is a difficult process. Accuracy depends on a complete understanding of how functions in the system being developed are met. Workload characteristics (frequency, volume, etc.) should be gathered in the current environment. Adjustments can then be made to estimate projected workload for the

future environment. Another means of estimating the size of a system to be developed is to examine the performance statistics of current systems of similar volume and architecture.

There are often various ways of supporting a user-required function; thus, the system configuration derived need not be unique. For instance, screen formatting as a user requirement can be translated into either high-speed transmission from the host or the satellite processor, or the use of intelligent terminals or local processors for screen formatting where less than high-speed transmission of data to the host computer will suffice. Each alternative should be evaluated according to how well it satisfies the highest priority requirements.

System requirements can be grouped into five categories: software, hardware, telecommunications, operating environment, and performance. A single user requirement can generate system requirements in more than one category as illustrated in the sample derivation of system requirements from user requirements shown in Figure 3.

Security considerations impose requirements on all of the above mentioned areas of the system. A risk analysis should be performed to determine the extent of the security provisions necessary to satisfy user requirements.

The following paragraphs outline some of the various areas of system requirements that must be considered in the computer selection process.

2.4.1 Software

Software support for the application being considered must be made in terms of the following considerations.

2.4.1.1 Operating System

The operating system must provide support in the following areas, where required:

1. Security and integrity--
   - Control of access,
   - Protection against simultaneous update when data is shared by many users;
2. Software products and utilities--
   - Software control utilities,
   - Application support packages, such as debugging aids, screen formatting software, etc.,
   - File system maintenance utilities, such as disk reorganization, data migration, and recall;
3. Interprocessors communications ability (where necessary)--
   - Online interface,
   - Job networking,
   - File transfer;
4. Appropriate file structures, including compatibility with interfacing systems;
5. Number of simultaneous users the software is capable of supporting.

2.4.1.2 Data Base Management System (DBMS)

If the need for a data base management system is warranted, its architecture must be evaluated to determine its suitability to the user's processing requirements for functional capabilities and efficiency.

2.4.1.3 Language

Systems that are designed using existing systems as a base have specific language support requirements. In addition, any system of this type also has a limited number of processors on which it can be used without a major conversion effort.

2.4.1.4 Automated Office System Requirements

These functions should be supported in office information systems: word processing, electronic mail, document storage, calendar planning, and typewriter-quality output.

2.4.2 Hardware

Hardware support must be made in terms of the following considerations.

2.4.2.1 Memory

Memory requirements are related to maximum execution program size (including code, tables, arrays, etc.) and operating system support (e.g., whether or not the system uses virtual memory).

User Requirement	Support in Area of	System Requirement
Ad hoc query of large data base	S	File management facility must provide random-access capability.
Simultaneous update by multiple users	S	Data protection at record level.
Multiuser online environment	P, S, H, T	Ability to support maximum number of users required efficiently.
High security	E, S	Control of access to ADP facility. Control of access to online/offline data.
Response requirement for processing of a financial transaction	P, S, H	Processor speed requirement derived from path length estimate. I/O requirements derived from estimated number of I/O's per transaction.
Building upon an existing system	S	Programming language and file structure compatibility of conversion tools.

KEY: S - Software
H - Hardware
P - Performance
T - Telecommunications
E - Environment

Figure 3. Sample Derivation of System Requirements From User Requirements

In virtual memory systems, the size of buffers and system tables that need to be resident in real memory or shared virtual memory is a consideration.

2.4.2.2 Accuracy

Number of digits of accuracy representable (most often governed by word size) is a deciding factor for applications with stringent accuracy requirements. Applications with numerical processing requirements should examine the arithmetic error-interrupt capability (e.g., overflow, division by zero, etc.) of potential computers.

2.4.2.3 Data Storage

Online storage required by an application consists of the components required by production and development work.

Storage for production work consists of the following components:

1. Production data bases--transaction files, audit files, master files, etc.;
2. Production program and procedure libraries.

Storage for development work consists of the following components:

1. Test data bases;
2. Test program and procedure libraries;
3. Storage taken up by programmer's private files, an estimate of which can be derived by taking the numerical product of the maximum number of programmers who have access to the system, the average number of files owned per user, and the average file size.

The total online disk storage requirement should be padded by a life-cycle growth factor.

The minimum number of tape drives required equals the maximum number of tapes used simultaneously by the application. If the application uses tapes that are imported from or exported to other systems, tape density is constrained by this factor.

2.4.2.4 Input/Output

Input/output requirements should be examined in terms of the following factors:

1. Input/output frequency and schedule;
2. Input/output volume;
3. Input/output media (form)--terminals, punched cards, graphics, microfiche, page printer, etc. (quality of output is sometimes a consideration, especially for office information systems);
4. Output distribution and utilization;
5. Input/output data protection.

2.4.3 Operating Environment

The operating environment must be considered with regard to the following factors:

1. Physical security and safety of hardware;
2. Geographic location of equipment, which impacts the serviceability and maintainability of parts, since some vendors do not have offices nationwide;
3. The operating environment, which must be understood when selecting terminals for an application. In some installations, many users have access to more than one type of terminal. The number and types of terminals to which the users already have access must be considered.

2.4.4 Telecommunications

Telecommunications applications should consider the following factors:

1. The number and type of terminals, their communications speed, and communications protocol must be specified;
2. The distance of transmission and the proximity of terminals, in order to determine whether or not the use of a cluster controller is cost-effective.

Applications that need to exchange data should be processed either on the same computer or on computers that have the appropriate interprocessor communications support facility.

Volume and speed of communications and their impact on total system performance must be evaluated by the system designer.

2.4.5 Workload and Performance

Workload should be specified in:

1. Number of runs processed per day;
2. Number of interactive sessions per day;
3. Hours of required operation;
4. Average and peak number of concurrent interactive sessions.

Performance specification should be stated in terms of peak load conditions. Peak workload should be specified in terms of the following considerations:

1. Maximum number of users;
2. Peak frequency of transactions;
3. Amount of processing for the average transaction in terms of lines of code (translated into number of machine instructions), number of operating system service calls (such as OPEN, CLOSE, etc.) amount of input/output (I/O) in terms of records read/written (translated into actual hardware I/O requests);
4. Growth in terms of number of users, number of transactions, and number of records processed per month.

2.5 Criteria for Evaluating Alternative Resources

2.5.1 Support of Existing Resources

Once the size of the application system has been determined, the current capacity of existing feasible equipment should be reviewed and an impact analysis made to see how the additional requirements will impact the overall performance of the equipment. This is most often achieved by modeling techniques.

2.5.2 Cost

Cost is often the bottom line in the selection of a computer resource for an ADP application. Cost estimates must be made on the system life-cycle basis.

The cost for computer system support consists of:

1. Hardware--processor, online storage, and peripherals;
2. Software;
3. Site preparation and space rental;
4. Maintenance--hardware and software;
5. Operator;
6. Environmental control;
7. System programmer where needed;
8. Materials and supplies;
9. Telecommunications service and equipment;
10. Equipment backup/redundancy;
11. Planning and procurement;
12. Application system support;
13. Training;
14. Management.

Cost factors vary from one computer resource to another. Besides the difference in purchase prices for a microcomputer, minicomputer, and large mainframe computer, the distribution of expenses varies with different types of equipment. Thus, cost-determining factors have been divided into three cost categories: mini/micro costs, in-house large mainframe and satellite processor costs, and teleprocessing service program (TSP) costs.

2.5.2.1 Cost of Mini/Micro Computer Support

When managed by the end user, the training and management costs for a system that uses mini/micro computers can be greater proportionately than those for the large computer environment because of lack of expertise in DP management. When users choose to operate their own computers, they must be aware of extra costs incurred due

to operation of the equipment and coordination required by the staff to accomplish the desired functions. They cannot rely on the data processing department to provide immediate assistance when support is needed in the total system operation (e.g., computer operators, data entry support, etc.). Users must spend more of their human resources to make effective use of the mini/micro computer.

The alternative to users providing their own support is for the data processing department to control the procurement, operation, and maintenance (or a subset of these functions) of the mini/micro installations along with the host and satellite processors. This alternative may prove to be most beneficial to all parties in the long run. Procurement and planning can be done by DP experts. When distributed processing is planned centrally, a more coherent growth can be achieved through standardization of hardware and software. (Even in a multivendor environment, there should be a standard interface with host and among satellite processors.) The DP department can absorb the cost of management of the processor better because of the economy of scale in the volume of work handled.

The feasibility of having the DP department responsible for the daily operations of the satellite processors depends on the complexity of the system and its geographical location. In view of the high cost of training, one consideration that should be made is the career growth path of persons working on satellite computers.

It is important to evaluate the security provisions of candidate systems. If a system does not offer sufficient support in this area, provisions added by the user can often be very costly.

The cost of hardware is normally low when compared to the total life-cycle support cost of the system. Users who assume that the cost of hardware represents the total cost of the system will find that they have severely underestimated the total cost.

2.5.2.2 Cost of Supporting a Production Application on a Large Processor

Most installations have a charge-back system whether or not real dollars are exchanged. Accounting algorithms vary greatly according to the values placed upon various services. When using the charge-back figures to compare the cost of alternative resources, one should make sure that the algorithms have as the basis that total recovered cost reflects the true cost of operations, including all of the items listed in the previous section, as well as the cost of auxiliary operations services such as keypunching, printing, plotting, etc. (OMB Circular A-121 requires that charge-back systems of Federal agencies account for the full cost recovery of ADP expenditures.)

2.5.2.3 Cost of Teleprocessing Services Program (TSP)

The full cost of TSP service can be derived by adding the following charges to the TSP bill:

1. Conversion;
2. Management;
3. Telecommunications service cost, where needed;
4. Application system support;
5. Training.

2.5.3 Other Considerations

If an analysis of user and system requirements indicates that neither existing in-house ADP resources nor resources available through TSP are appropriate, the acquisition of ADP equipment will be required. The need for a dedicated processor for an application must be justified on a cost-effective basis.

The selection of ADP equipment will be based on system requirements as well as the following considerations.

2.5.3.1 Compatibility with Existing In-House Hardware and Software

The compatibility of hardware resources increases the flexibility of configuration. The compatibility of software reduces the personnel cost of supporting multiple systems. Compatibility with existing in-house ADP resources protects the previous investments in software/hardware and personnel training. Availability of common user facilities such as terminals and remote job entry (RJE) should also be taken into consideration.

2.5.3.2 Ease of System Expansion to Meet Future Requirements

Since the personnel cost involved in a major procurement is high, and the process of a conversion is resource consuming, the ability to upgrade ADP equipment should be weighted heavily in the selection process.

2.5.3.3 Vendor Support

The cost of system unavailability in terms of management cost and degradation of programmer/user productivity should be taken into consideration in comparing vendor-quoted or published mean-time-between-failure (MTBF) and mean-time-to-repair (MTTR) statistics.

2.5.3.4 Hardware and Software Maintainability

Geographical location and level of personnel (in-house or vendor) expertise contribute to hardware and software maintainability.

2.5.3.5 Time Constraints

Procurement, installation, and training time must be considered relative to the urgency of users' needs.

2.6 Summary of the Computer Resource Selection Process

Figure 4 summarizes the resource selection process. This process consists of the following steps:

1. Collection of user requirements;

2. Analysis of user requirements. The suitable class of equipment can be identified in most cases by using the principles stated in the data processing department's long range plans as baseline resource selection criteria. In instances where user requirements do not clearly fit any of the blocks identified in Figure 4, subsequent steps of analysis should be taken for all possible configurations. There are also applications for which the processing can be distributed across computers (e.g., data entry performed on a minicomputer and data base management operations performed on the host); separate analysis should be done for both components of the system;

3. Derivation of system resource requirements. Once the alternate resources are narrowed down, the size of the application (in terms of the amount of physical resources necessary to support it) can be determined for each alternative class of equipment;

4. If resource identified in second step is available in-house, assessment of the total impact of supporting an additional application to determine if there is sufficient capability, appropriate software, sufficient support personnel, and a requirement to change current operating procedures. Results of the impact study may lead to the procurement actions for TSP service, hardware/software, or acquisition of support personnel;

5. If resource identified in second step is not available in-house, searching for a solution based on the Teleprocessing Services Program (TSP). If no appropriate resource exists there, procurement actions for equipment of the class identified must be initiated. In addition to the system requirements, the factors discussed in Section 2.5 of this procedure (other than system requirements) should be considered.

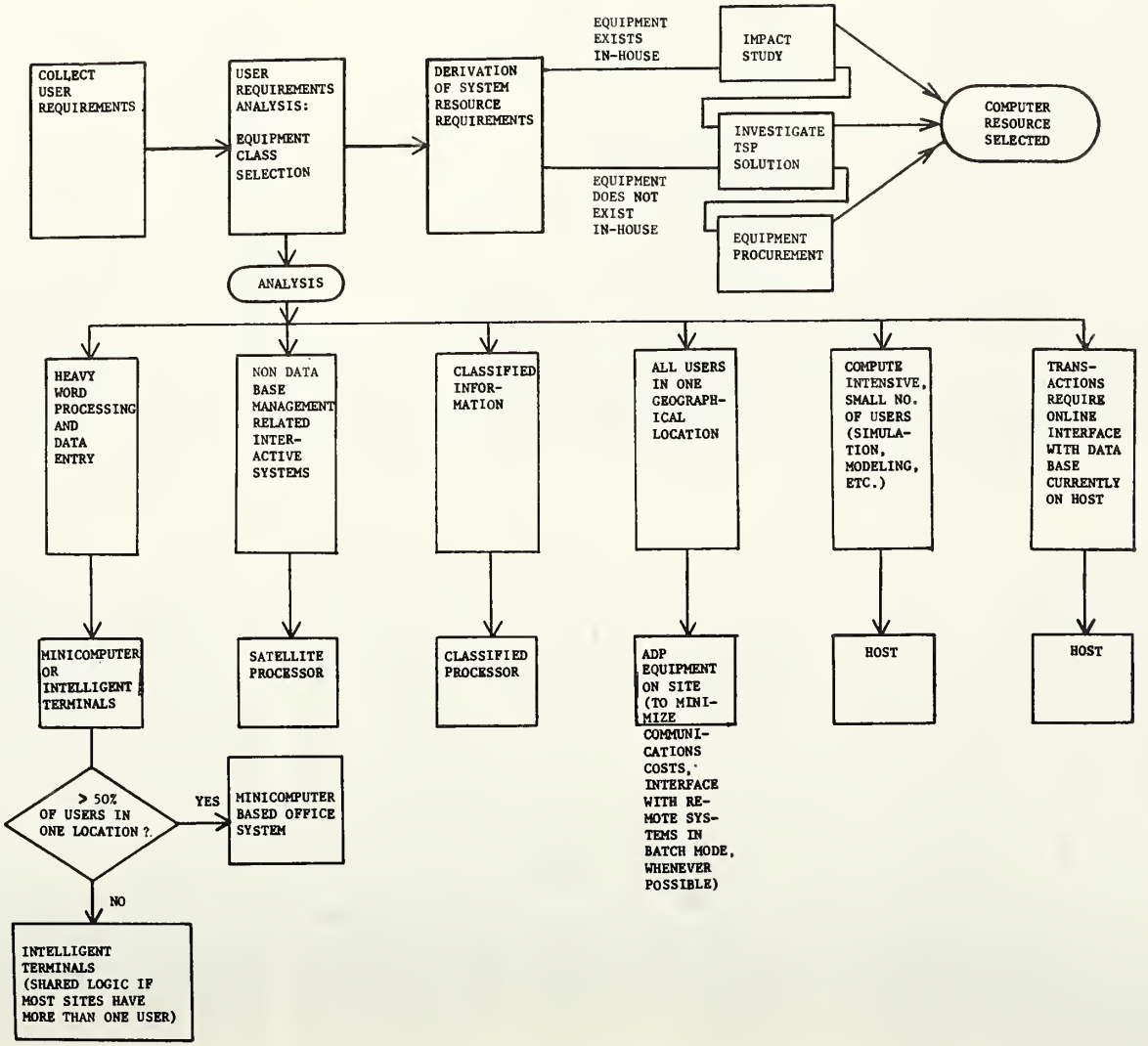


Figure 4. Resource Selection Process

CPEUG81

Panel Overviews

OFFICE AUTOMATION AND PRODUCTIVITY

MARY L. CUNNINGHAM

NATIONAL ARCHIVES AND RECORDS SERVICE
OFFICE OF RECORDS AND INFORMATION MANAGEMENT
WASHINGTON, DC 20408

Productivity is a very real problem in the Federal Government. Many Federal agencies are facing increased demands for services at a time when resources are being reduced. To meet conflicting demands we must find new ways to do our jobs better. That means, among other things, that we must improve information management. Managers of offices are managers of information. Managers are effective in running their various programs only to the extent that they manage the information upon which the programs depend. Needed information must exist in the most convenient place, at the time when it is needed, and in a form that can be used most efficiently.

The most important development in recent years in the field of information management has been the rapid advance--you can even call it a revolution--in information technology.

"Word processing," basically linking high speed electric typewriters with computer capacity, has been part of this revolution. The impact on the Federal Government has been dramatic. In 1979 the General Accounting Office reported that in fiscal 1977 Federal agencies acquired word processing equipment that cost about $80 million. This same report estimated that by this year the annual expenditure for acquiring new word processing equipment would be $300 million. A generally declining price structure for word processing equipment, combined with increasing pressure for greater efficiency and productivity in written communications, virtually guarantees that by the mid-1980's word processing will be a major growth area. But the fast-moving technology keeps taking new directions. Both Government agencies and private businesses that traditionally produced and acquired information in paper form are turning more and more to other media. Undoubtedly this trend will continue, and even accelerate, as the technologies become more advanced, more effective, and less expensive.

Computer professionals are certainly aware that computers are becoming smaller and cheaper and are being used more widely. They will eventually reach into virtually every office, and into every aspect of information creation, maintenance, and use. With new, simple-to-use minicomputers, ADP support will be used by more and more non-technical professionals and managers. Through the use of communications technologies, access to information will dramatically increase.

Another important trend that must be recognized is the merging of information technologies. Today a single terminal may be a word processor, a data processor, a link in an electronic communications network, or all of these. According to a recent Washington Post article by Will Sparks, Vice President of Citi-Bank Corporation, "All information traveling in the world's information stream is becoming an homogenous flow of digital bits,...and since, in many cases, computation is being performed on these bits of information while they are in transit, the distinction between computing and communicating--or between the computer and the telephone--is vanishing."

All of these trends are of immediate concern to Federal managers. By and large, the ordinary Federal manager is ignorant of the intricacies of computer science and communications technology. They can banter the jargon, a little, but must depend on computer professionals to tell them how it all really works. Still, because it's their office and their mission, they must ensure that the technology used is cost effective and that it achieves its potential for controlling the information explosion and increasing worker productivity.

This aspect of management goes beyond the documentation of requirements and the establishment of performance criteria before installing a system. Large, technically complex systems that affect

many office processes and procedures may require considerable expertise to understand and operate effectively. Even the smaller, single-function systems--such as micrographic systems for records storage and retrieval--require the continuing attention of the office manager if they are to produce the desired improvements in office performance.

Effective management today, then, requires not only effective managers of information, but also effective management of the technological tools of information management. Non-technical program managers must understand and effectively use office automation. Computer science professionals will have to be patient with those managers and teach them what they need to know. They in turn, must be more careful to identify and explain what they are trying to accomplish. This is not an easy task. To date there is not even a widely accepted definition of office automation, much less a set of valid standards and guidelines to use in decision making. Office automation is sometimes defined in terms of the equipment used rather than the goals or objectives it is installed to fulfill. Popular descriptions of the "office of the future" represent the tasks of the office as being the same from organization to organization. That view focuses on the form of the work in an office, not its substance. In that view, offices exist to produce documents, to communicate, or to store and retrieve information. If that popular view of office automation prevails, then office automation will not be cost effective nor achieve the goal of increasing overall productivity. "Improvements," or changes in form alone can rarely justify the expense of installing these technologies. To be effective--to make real improvements in office efficiency and productivity--an office automation system must be carefully designed to support the real mission of that office. It must be designed with the goal, not the process of the office in mind.

It is easy, as historians of the American experience have recognized, to get carried away by exciting advances in technology. We only have to think back to those days in the 1840's when Samuel Morse's new electric telegraph had stirred American imaginations and entrepreneurs quickly began to wire the country.

One historian of the telegraph--Robert Luther Thompson--has labelled the 1846-1850 period as the time of "methodless enthusiasm." Everybody wanted to get into the act. Hastily, poorly built and duplicating systems were strung throughout the East and into the Midwest as the result of starry-eyed dreams of quick profits rather than in response to needs and carefully considered plans to fulfill them.

This was the period of which Henry David Thoreau wrote in Walden: "We are in great haste to construct a magnetic telegraph from Maine to Texas, but Maine and Texas, it may be, have nothing important to communicate."

The telegraph was one of the great developments of Nineteenth century technology, but its commerical beginnings were notable for waste, misapplications, and failed telegraph companies. One of the best lessons of that experience of "methodless enthusiasm" was that you must think in terms of needs and goals when employing new technologies, rather than seize upon them for unsuitable purposes just because they are "there." Let's hope we remember those lessens as we once again get "wired".

A good place to start a discussion of new technologies is with definitions. Analysts at the Massachusetts Institute of Technology have defined office automation as: "The utilization of technology to improve the realization of office functions," or, rather, the business objectives of an office. This definition stresses increased productivity in the achievement of specific end results.

Seen in this way, the principal benefit of office automation is the increased productivity that results when employees can perform necessary tasks more efficiently. The objective of office automation, then, is not to eliminate paper--and certainly not to produce more paper faster--but rather to help us do the work we need to do, faster and better.

Given the potential for savings, it is understandable that many agencies are interested in office automation, and many cooperative, interagency efforts have been initiated to develop a coordinated approach to it.

Over the past year, for example, GSA, OMB, and OPM have cooperated on a project to assess the use of office automation by Federal agencies. Most of the systems studied during the project were designed for information storage and retrieval, for text editing, or for data analysis, using data processing, data transmission, and word processing technologies.

Many were designed to perform several functions, using more than one technology. All have as their objective improved delivery of public services and better management of Federal programs. For example, in the area of improved delivery of services, the Social Security Administration now uses a telecommunications system to perform necessary files review. The system has significantly reduced the length of time that needy beneficiaries must wait for their checks, and is expected to save the taxpayers about $25 million between FY 1980 and FY 1984. A new Veterans Administration system, called TARGET, is expected to reduce regional offices' staffing by 1,488 work-years over the next 3 years. And, of course, the processing of income tax returns already benefits from office automation. A new IRS system rapidly checks the accuracy and currency of taxpayer identification data, so watch out.

This survey demonstrated that Federal managers are designing and installing a wide variety of office automation systems to perform many different tasks. It demonstrated, further, that there is considerable potential for savings when office automation technology is used appropriately. But we must emphasize that those systems must be installed carefully. Social Security's system, for example, was the result of more than 2 years of system design by a large number of Federal and contractor personnel.

Unfortunately, managers too quick to opt for technological solutions to systemic or procedural problems often find that the results are disppointing. The GAO report cited earlier said that failure to achieve expected savings from the use of word processing equipment was the rule rather than the exception throughout the Federal Government. It has been our experience that a thorough systems analysis is necessary to give managers a solid basis for making informed decisions about what to expect from specific applications of office automation technology. Office automation is no magic solution to such problems as poor clerical skills, backlogs, or poor turnaround time.

We are now in the process of documenting the characteristics of systems and operations having high potential for the cost-effective use of office automation. The standards we expect to develop, based on this research, will cover both the kinds of applications that can benefit from office automation, and the methods for studying and evaluating the potential of individual applications. We plan to study the experiences of the office managers, to learn lessons they have learned, and to share them with others.

Our goal is to help program managers identify and meet their needs. We intend to provide the guidance managers will need to understand what office automation can-- and cannot-- do for them. Managers must first of all understand the unique features and capabilities of different types of equipment. Secondly, they must be able to identify good applications for this equipment in their own organization.

Finally, they, and we, need to be reminded that office automation can't do it all. We hope that our cooperative efforts over the next year will lead to better identification and documentation of the requirements for designing and implementing effective automated systems in the office.

COMMAND, CONTROL, AND COMMUNICATIONS MANAGEMENT CHALLENGES IN THE 80'S

Robert G. Van Houten
HQ Army Command & Control Support Agency

The four speakers in this panel will address a few of the many different aspects to command, control and communications.

Panel Members are:

Mr. Joseph Volpe

Modernization of the ADP and supporting data communications for the World Wide Military Communications and Control System (WWMCCS) taking place under the WWMCCS Information Systems (WIS) Program will be discussed. A description of the historical perspective associated with WWMCCS ADP and its status today will be presented. The WIS modernization goals and problems will be covered briefly with the associated WIS architectural solutions. Some challenges will be presented which reflect the maintenance automated test (bit/byte) needs which stem from all levels of modern warfare as projected into the 1990's.

Col. D. L. Moore

Discussion of Computer Performance Management (CPM) requirements and activities for the NORAD/ADC operations systems (Program 427M). This will include a brief description of the NORAD/ADC operational ADP systems and will cover the CPM program developments, accomplishments to date, including problems encounted and lessons learned, current status and activities of the CPM program and a proposed approached for the foreseeable future.

Col. Joe Newman

Opportunities for performance management of automation and communications networks for the Worldwide Military Communications and Control System (WWMCCS) at three levels: macro level, intermediate level, and the micro level will be presented.

Dr. William Wenker

Communications management and capacity planning problems and goals associated with the WWMCCS system architectures in the 80's will be presented..

ADP COST ACCOUNTING & CHARGEBACK

Chairperson: Dennis M. Conti
National Bureau of Standards

Panelists: Kenneth W. Giese
FEDSIM

Christos N. Kyriazi
Department of Commerce

Ranvir Trehan
MITRE Corp.

ABSTRACT

There is no question that the ADP shop plays a vital role in the day-to-day operations and survival of most organizations. However, two distinct views still exist on the nature of the ADP shop within an organization. One view exists that the ADP shop is an overhead function that should provide free service to the users. Another view sees the ADP shop as a service center that should run on a profit and loss basis. An extension of this latter view is that the full costs of ADP should be accounted for and charged back to the users. However, whether or not these charges are actually recovered from the users, the mere reporting to the users of their fair share of the total ADP costs is believed to have a beneficial effect by increasing the users' awareness of and accountability for their use of ADP. Most recently within the Federal Government, OMB circular A-121 requires agencies to implement full cost accounting, and where appropriate, cost recovery. Examples of some of the costs to be accounted for under A-121 are software depreciation and space occupancy costs, in addition to the more traditional ADP cost elements.

Although many sites currently have some form of chargeback system, few account for all of the costs of ADP. For those organizations that are considering implementing a chargeback system, reluctance by both the ADP manager and the user community to make their activities more visible makes such an implementation more difficult.

The panel will address several specific aspects of cost accounting and chargeback. Questions that the panel members will be asked to address include the following:

1. Is the expense of implementing a full cost accounting and recovery system justified by its benefits?

2. Is the actual recovery of costs from the users necessary, or will "information reports" suffice? What are the advantages to be gained by an actual transfer of funds?

3. What are the salient features of a good chargeback system?

4. What are some of the human/political problems associated with introducing a chargeback system into an organization?

5. How does the trend toward distributed systems complicate the cost accounting and chargeback process?

The panel contains individuals with experience in both developing chargeback guidelines, as well as overseeing the implementation of chargeback strategies in a large organization.

SUPERCOMPUTERS, CPE, AND THE 80'S
PERSPECTIVES AND CHALLENGES

F. Brett Berlin

Cray Research, Inc.
1919 Pennsylvania Ave., NW
Washington, DC 20006

The public glamour of minicomputers, microcomputers, and their derivatives (e.g., "distributed systems", "personal computers," "paperless offices," etc.) tends to support the misconception of supercomputing as a soon-to-pass anachronism of use to few beyond the specialized worlds of weapons, nuclear power, or weather prediction and research. Recent events in the supercomputer marketplace, however, indicate that true supercomputing is only in its embryonic stages and that supercomputers will become an even more crucial segment of the CPE practitioner's challenge. While actual numbers of installed systems are still relatively low -- less than 40 worldwide, including all vendors -- the breadth of users is rapidly extending far beyond the walls of the national research laboratories such as Los Alamos and Lawrence Livermore Labs. The Cray-1S supercomputer, for example, is installed at five scientific service bureaus, several universities, and a few commercial industrial accounts. Even more significantly, the number of identified supercomputer prospects has grown from less than 80 in 1972 to over 300 today. As IBM, Hitachi, and Fujitsu realize already announced intentions to build supercomputer-class systems by the mid-80's, the CPE challenges now faced by only a small coterie of supercomputer users will extend to most sectors of the community. The primary objective of this seminar is to identify these challenges and to discuss directions of supercomputer technologies for the mid and late 80's.

During this session, panelists will discuss their current work in the supercomputer field and will identify the key current and future issues which affect supercomputer development. Some of the major issues to be discussed include:

1. Major CPE Challenges to Current Supercomputer Users: The CPE practitioner is faced with the same "classic" questions common to large system, but the unique architectures and performance characteristics may make common CPE tools of only marginal value, and previous results from conventional architectures useful as a baseline only.

2. Problem Architectures vs. Supercomputer Architectures: Much of the speed advantage in a supercomputer results from parallel architectures. These can be based upon pipelines, processor arrays, or parallel processors. However, current designs are seriously constrained in usefulness by "problem architectures"--or the way the problem and algorithm is designed to take advantage of certain architectures. There are significant costs to tailoring problem software to meet special architectures, both in conversion and future transportability, but there are some cases in which these costs may be reasonable when compared to potential performance savings.

3. Vendor-developed Software Tradeoffs: "Main line" vendors, such as IBM, DEC, etc., have developed software-rich environments in which most users rely heavily upon the vendor for the majority of their system software and utilities needs. Standard packages abound for almost every application, allowing virtually any user to "computerize" without sophisticated in-house development. Supercomputer vendors and users alike are faced with a difficult set of questions which determine the desireability of development and support of software functions expected by other market sectors. One such function is virtual memory, which has been implemented by one supercomputer vendor but rejected by another as detrimental to optimum performance.

4. Directions of Future Architectures: Supercomputer researchers and manufacturers expect to achieve dramatic performance goals before 1990. In order to accomplish these goals, developers must make certain assumptions concerning both technology and the future application environment.

SESSION PANELISTS

William Alexander

Los Alamos National Laboratories
Computer Division
Los Alamos, NM 87545

Richard McHugh

Cray Laboratories, Inc.
3375 Mitchell Lane
Boulder, CO 80301

Art Lazanoff

Control Data Corporation
8100 34th Avenue South
Minniapolis, MN 55440

Jack Worlton

Los Alamos National Laboratories
Computer Division
Los Alamos, NM 87545

ADP CHALLENGES IN MEDICAL, PERSONNEL, AND WELFARE SYSTEMS

Chairperson: DINESH KUMAR
Social Security Administration

Panelists:

Dr. Joe H. Ward, Jr.
Air Force Human Resources Laboratory

Mr. Paul R. Croll
Office of Personnel Management

Lieutenant Colonel Joe Ribotto, Jr.
TRIMIS Program Office

Colonel George A. McCall
Air Force Manpower and Personnel Center

Mr. Dave England
Texas Department of Human Resources

Mr. Norman Clausen
Internal Revenue Service

This session highlights the needs, constraints and opportunities associated with automatic data processing applications in the fields of medical, personnel and welfare systems. The state-of-the-art approaches that are being undertaken by senior managers involved in the design, development and enhancements of the ADP system will be presented. The difficulties encountered in planning, consolidation and acquisition of computer and communication systems will be explored. This session includes several planned presentations by the panel members followed by questions and comments from the audience.

The panelist presentation in the area of medical systems will concentrate on the discussion of ADP systems and approaches being undertaken in the area of medical management information systems. The functional capabilities of these systems will be described along with their status and difficulties encountered during implementation.

In the area of personnel systems, the planned presentations will cover the latest concepts in planning, design and research on personnel action systems. The procedures for the implementation of personnel policy through computer based systems will be discussed. The policy generated models result in pay-off (utility) of various person-job actions. The techniques for generating alternative pay-offs and their optimization will be discussed. The current and future plans for enhancements of an existing manpower and personnel data system will be discussed to illustrate the phased implementation of research concepts. Reference will be made to the operational Procurement Management Information System (PMIS). The other planned presentation will cover the automation opportunities in conducting tests/examinations along with related psychometric considerations.

The difficulties encountered in planning and consolidation of ADP requirements and future needs, due to several independently run data processing centers, (as compared to a primarily centralized data processing operation) will be highlighted.

In the area of welfare systems, emphasis will be on the difficulties experienced in managing large ADP and Telecommunications systems along with a discussion of constraints faced by various installations.

INFORMATION SYSTEMS AND PRODUCTIVITY IN STATE & LOCAL GOVERNMENT

Ron Cornelison

Office of Administration
State of Missouri

Persons attending this session will gain an understanding of the problems and opportunities present in the public sector at the state and local level relating to applying technology to achieve organizational objectives. The three branches of government--Executive, Legislative and Judicial--will be represented on the panel. The panelists are: Mr. Christopher Burpo, Arthur Young and Company, San Antonio, Texas; Mr. Steven Claggett, Regional Justice Information Service, St. Louis, Missouri; Mr. Ron Cornelison, Missouri Division of EDP Coordination, Jefferson City, Missouri; Mr. J. C. Humphrey, Texas Legislative Council, Austin, Texas; and Ms. Carolyn Steidley, Missouri State Court Administrator's Office, Jefferson City, Missouri.

The panelists will discuss the missions of their respective governmental agencies and how automation has been applied to increase organizational productivity. Questions and comments from persons attending will be welcome in this session.

THE ADP PROCUREMENT PROCESS - LESSONS LEARNED

Terry D. Miller

Government Sales Consultants, Inc.
Annandale, VA 22003

The purpose of this session is to review for listeners the latest techniques in ADP competition, the mistakes that others have made, the latest rules, etc., so that we will not reinvent the wheel.

Colonel Donald W. Sawyer or a member of his staff will discuss the Air Force study on how to improve the procurement process by assisting field offices on the procurement learning curve.

Steve Lazerowich, Federal Proposal Manager of Digital Equipment Corporation in Lanham, will discuss how a mini vendor looks at government procurements.

I will discuss the latest regulation changes and provide continuity and a wrap-up.

I am not sure you can say we will advance the state-of-the-art. In the procurement business most people are still learning how to walk. We will provide detailed lessons on how to walk without falling into snakepits.

DATABASE MACHINES

Panel discussion chaired by:
Carol B. Wilson

Fiscal Associates, Inc.
Alexandria, VA 22304

Over the past 10 years, the use of data base management systems has been growing as the need to effectively manage and display data has increased. This use of data base management systems has come about because they offer several advantages: functional capability, online query capability, and reduced manpower. A DBMS affords the user such things as data protection, concurrency control, audit trails, and backup/recovery features, which he does not have to program or design himself. Likewise, manufacturers of DBMS's have developed sophisticated, easy-to-use online query capabilities which permit the user to peruse his data easily, and with no programmer intervention. DBMS's help to reduce manpower requirements through the provisions of data independence, already programmed data management procedures, flexibility in changing data base design, and the ability to quickly respond to ad hoc inquiries.

Unfortunately, these attractive advantages do not come without a price. Most data base management systems suffer in both the performance and cost areas. They often place extraordinary requirements on the hardware because they are large and complex and because they operate on general purpose operating systems and on general purpose architectures. Cost is also a factor in the DBMS environment as more hardware is purchased to accommodate the increased resource demands. Hidden costs may also arise from service and performance problems affecting other, non-DBMS users of the ADP installation. The performance problems brought about by the extraordinary requirements on the hardware and operating system can, and do, lead to decreases in ADP installation productivity. Another problem in the Federal sector comes about because many DBMS's are dependent upon the hardware and operating system. The lack of machine independence leads to either less competition in future procurements, or in expensive conversions.

The advent of data base machines could possibly be a solution to the problems mentioned above: decreased ADP installation productivity and hardware independence. The data base machine is a backend system with both software and hardware specifically designed for efficient data base operations. The resource demands normally placed on the ADP system for data manipulation are off-loaded to the data base machine. Changes in mainframes may be accommodated by the development of interfaces from the main system to the data base machine which would afford some hardware independence to the data base system.

The panelists will discuss why and how the productivity of an ADP installation could improve through the use of data base machines. They will describe the communications interface and the unique architectural features.

CPEUG81

Tutorial Overviews

MEASUREMENT, MODELING, AND CAPACITY PLANNING : THE SECONDARY STORAGE OCCUPANCY ISSUE

H. Pat Artis

Morino Associates
Vienna, VA 22180

During the past decade, the economics of computer performance have undergone a profound transition. Although hardware costs were dominated by the central processor in the early 1970s, today the relative costs of the central processor and secondary storage are approximately equal for most large installations. An example of this cost transition is shown for a FORTUNE 50 corporation in Table 1.

	1970	1980	Percent Change
CPU Configuration	360/65J 360/40F	3033U8 3032U4	
CPU MIPS	0.8	7.0	875
DASD Configuration	16 2314s	128 3350s	
Bytes (billions)	0.46	42.0	9150
MRC Expense			
CPU	$66,060	$134,400	203
DISK	$12,360	$108,800	880
Ratio DISK/CPU	0.19	0.79	420
Floor Space	25,000	80,000	320
CPU	6,000	9,000	150
DISK	5,000	40,000	800

Table 1: Typical Installation Growth 1970 - 1980

In 1970, the monthly rental cost (MRC) of the corporation's CPUs was six times greater than the cost of secondary storage. Today, these costs are approximately equal. Although the corporation has witnessed a growth in CPU power of 880 percent, their secondary storage space has grown by more than 9,150 percent to 46 billion bytes. Moreover, secondary storage now represents the majority of the floor space, power and cooling costs for the installation.

For users of smaller systems like the IBM 4331 or the IBM 4341, the cost transition has been even more dramatic. It is not uncommon for the secondary storage costs of a 4300 class system to be two or three times the processor cost. Clearly, computer performance evaluation (CPE) must address the issue of secondary storage occupancy to control the costs of data processing in the 1980s.

CPE in the 1970s has primarily concentrated on the measurement and modeling of the I/O traffic between the central processor and the secondary storage devices. Unfortunately, very little effort has been expended on understanding how efficiently the space on these devices is being used. This tutorial addresses the following topics relevant to secondary storage occupancy issue:

MEASUREMENT. The available data and design considerations for a secondary storage occupancy measurement scheme will be reviewed. Applications of data from a prototype system developed by the author will also be discussed.

SPACE MANAGEMENT. In the past several years, a number of secondary storage management systems (HSM, DMS/OS, ASM2, ... etc.) have evolved that allow installations to automatically archive datasets that have been unreferenced for a specified number of days. Modeling techniques that allow users to predict the behavior of a secondary storage management system in their installation and to judiciously select the archiving criteria will be discussed.

DEVICE ARCHITECTURE. Although the most common statistic quoted by the vendors about secondary storage devices is that the cost per byte has steadily declined, other statistics are somewhat more valuable. Beginning with the announcement of the double density 2314, the 3330, 3330-II, 3350, and 3380 have represented a steady decline of the accessibility per byte (i.e., maximum I/O power divided by device capacity). In many installations, it is not uncommon to find that only thirty or forty percent of the datasets on the busiest packs are accessed even once per week. Therefore, any attempt to effectively use the remaining space on these devices may result in unacceptable I/O scheduling bottlenecks. This problem will require the development of dataset placement algorithms that attempt to optimize an entire installation rather than a single pack.

CAPACITY PLANNING. An approach to DASD capacity planning that incorporates device occupancy, performance and architecture considerations will be discussed. This approach is currently being evaluated by the author.

The audience will also be invited to propose other topics for discussion.

MICRO-COMPUTERS IN OFFICE AUTOMATION

Wendell W. Berry

National Oceanic and Atmospheric Administration
Office of Budget and Resource Management
Rockville, MD

Key words: micro-computers, office automation.

This tutorial describes procedures followed in office automation projects and emphasizes the significant part played therein by micro-computers. Applications are those experienced over a six-year period of involvement in administrative automation within the National Oceanic and Atmospheric Administration, (NOAA) of the U.S. Department of Commerce.

The administrative support environment of a service oriented agency broadly defines the scope of automation projects discussed. Criteria were developed to guide the institution of and measure effectiveness of such projects. A systematic approach was adopted, and began by defining functions of the specific target office.

Office functions and tools and procedures used to fulfill them were of immediate concern. These helped us in forming an answer to the most important question - who was doing the work, what skills and attitudes toward changes system changes were present. Most functions were found to have important inter-organizational aspects to them. A basic management tool became the categorization of functions into "administrative processes" that together define the management of resources in the organization.

Finance, personnel, facilities, material, and general administrative services were five major groups defined. Administrative divisions and technical operating units contained portions of these processes. A high degree of inter-relationship among the groups also existed, and the integrated automation of all of them could be described as the ultimate office automation goal.

Within the macro structure that provided a dynamic system definition, individual objectives were defined. Minimal system disruption and cost savings/avoidance were often sited by those initiating such projects. Tasks were scheduled based upon opportunity, politics, and an overall attempt to develop basic system modules as a primary priority.

Two basic methods of automation emerged. Integration of existing data into routine work procedures was the first. Data found in existing batch systems formed this basis. Automation of procedures not previously economical was the second.

The necessity of micro-computers in the above described scenario becomes clearer upon consideration of their application. In their role as "personal" computers, micros provide an interface between humans and the power of computers, essential to the further development of office automation. Following is an introduction into some areas of micro-computer uses in administrative computing.

Information Processing

1) Word Processing - As price reductions and market competition begin to bring the multi-purpose micro into the office and the home, word-processing is becoming a specialized software application, as opposed to a problem with a hardware dominated solution. Word-processing is now introduced to many as a by-product of the acquisition of a micro-computer system, rather than as an end unto itself.

2) Data-Entry - Exotic forms of rapid data entry are presently utilized in large volume applications, where their cost may be amortized over many thousands of source documents. For other systems, many key-punch operations still continue, and much data is also keyed in an on-line environment into a mini-or maxi computer. The dis-economy of using computers at human speed provide yet another opportunity for efficent use of the micro-computer as a "front-end" device to other computers.

Information communication

1) Data Transmission - Within the last few years, much research and development work has been concentrated upon communication systems enhancements. The use of micro-computers as part of an information communication network has emerged as result of this emphasis.

2) Management Information Systems - Interactive graphics has been made available at an individual managers level on a practical basis by the micro-computer. The much talked about "what-if" questions may by quickly and vividly answered by graphics, and most recently, in color.

3) Electronic Mail - Electronic mail networks now exist on many mini and maxi computers. The proliferation of words in magnetic form allow micro's to perform the function of a personal electronic filing cabinet. Off-line information storage on inexpensive media is required for large-scale communications via computer.

Self-Contained Systems

1) General Development - Shared use of micro-systems has caused many small manual systems to be automated. BASIC remains the dominant micro language, with many "dialects" in use. Pascal, Fortran and Cobol also compete for system use.

2) Micro DBMS - An answer to the language conflict and also a part of industry-wide empahsis upon DBMS', micro DBMS are now emerging. Non-data processing professionals now follow a systematic approach to development of their systems.

Current Developments

1) Human Engineering - Error detection produces perhaps 40% of the total amount of program coding in many applications. Reducing screen scrolling and providing audio and flashing screen error indications are examples of features designed with the end user in mind.

2) Increased computational power - More eight-bit micro's are now in use than all other types of machines combined. Sixteen bit machines are now in many product lines, and thirty-two bit chips are a reality. These will put mini-computer power on desktops within a short period of time.

3) Increased storage - Hard disks have arrived for micro use, bringing with them capacities measured in millions of bytes. Implications for future expansion of micro usage are wide ranging, considering their projected capabilities.

4) New designs - Microcode implementation in hardware to support the ADA language, and the delegation of interrupt handling to subsidiary 16-bit processors in the new 32-bit chips are an example of a significant departure from traditional architecture.

COMPUTER PERFORMANCE MANAGEMENT IN THE ADP SYSTEM LIFE CYCLE

James G. Sprung

The MITRE Corporation
McLean, VA 22102

During the last ten to fifteen years, computer performance has evolved into a sophisticated manager-analyst interactive process. The reason for this is the recognition by government ADP managers, as well as non-government managers, that a need exists for doing computer performance throughout the ADP system life cycle. The manager continuously faces requests from users and oversight agencies for computer resource availability data.

The goal of this tutorial is to provide the framework for a continuing computer performance management program throughout the ADP System Life Cycle. The tutorial will include descriptions of:

1) The ADP System Life Cycle,

2) the types of performance studies, and

3) the types of performance tools

The different tools will be described along with how they can be used to support the performance studies. The performance studies will be described along with how they should be used throughout the ADP System Life Cycle. Throughout the tutorial, examples will be presented that describe the use of each of the tools for different types of studies during each phase of the life cycle.

Computer Performance Management must continuously be of service to managers to be valuable as an agency function. Therefore, the need of the CPE analyst to both perform studies and present meaningful results is critical. The key to the continuous aspect of the computer performance function is the analysts ability to recognize when the next stage of the life cycle is occuring and to perform the studies that are appropriate at that time. Therefore, the computer performance staff needs to serve both a planning and monitoring function. The ADP resource, to most agencies, is critical to the agencies existence and with the help of a computer performance staff, the ADP resource can provide adequate capacity to meet the agencies needs.

SIMULATION TOOLS IN PERFORMANCE EVALUATION

Doug Neuse
K. Mani Chandy
Jay Misra
Robert Berry

Information Research Associates
911 West 29th Street
Austin, TX 78705

The selection of modeling tools for performance evaluation is discussed. Simulation and analytic tools are discussed and compared. A simulation system designed to accept computer readable, picture-oriented models is described.

Key words: Analysis; performance evaluation; pictures; simulation; tools.

1. Introduction

This paper is concerned with the question of selecting the modeling tool most appropriate to a performance evaluator's problems. We first compare simulation and analytic methods. The advantages of each method are discussed. We next study two common types of problems faced by performance evaluators, and describe the method of choice (simulation or analysis) for each type. The important characteristics of good performance tools are discussed next. The importance of ease of input to the modeling system is emphasized. A case is made for using pictorial or diagrammtic representations of computer systems as the input to simulation systems. A simulation system that is designed to accept such (computer readable) pictorial descriptions is described.

2. Modeling Methods: Simulation Versus Analysis

There are two approaches to the modeling of computer systems: simulation and analytic. Analytic models consist of sets of equations, usually derived from queueing theory. Discrete-event simulation models simulate the generation of transactions and the processing of transactions over time. The tradeoffs between simulation and analysis are described next.

2.1 Advantages of Simulation

2.1.1 The simulation methodology has been validated over and over again. There is absolutely no question about the power of the simulation approach. Given enough time and enough measured data, it is possible to simulate any system to any required degree of fidelity.

2.1.2 In contrast, the analytic approach is limited in power: there are no general analytic techniques that apply to every situation. Different sets of equations have been developed to model different systems. When a set of equations yields, for one system, predictions proven to be correct, our faith in the ability of that one set of equations to model the given system increases; however, our faith in the ability to model all systems by similar sets of equations should not increase. There is no way of validating a general analytic methodology because there is no general analytic methodology. Each analytic model is usually a distinct set of equations. At this time, there is no

tractable method to generate a set of equations to model any given system faithfully. There are classes of analytic models, such as Markov models, that are very general. However, such models are not always tractable because the number of equations in a model may be too large to be solved in a reasonable amount of time.

2.1.3 Simulation is easily understood by programmers. It is much easier to hire programmers with some experience in simulation than it is to hire analysts with expertise in analytic models.

2.1.4 Simulation is flexible. A simulation model can always be changed to track changes in the system being modeled. An analytic model may work well for a system initially, but poorly when the system is changed. It is not easy to change a set of equations to model the effect of a new scheduling policy, for instance.

2.2 Disadvantages of Simulation

2.2.1 Simulations require much more computer time than analytic models.

2.2.2 There may be a paucity of measurement data from the system to be modeled. If we do not have the measurements to build a detailed model, there is no point in building a detailed model. Analytic methods may work as well as simulation for simple models. Simulation models are justified when the data to parameterize simulation models exists.

3. The Method of Choice

The method of choice -- simulation, analysis, or a hybrid approach combining simulation and analysis -- depends upon the problem being solved. Consider different situations in which models are used.

3.1 The Design of New Systems

Important characteristics of design efforts include the following.

3.1.1 Designs often change before they are implemented. Models must be able to track rapid, and sometimes radical changes in designs.

3.1.2 The model must be understood by the designers. This point is very important. If designers do not understand a model, they are less likely to cooperate in the modeling effort. Active cooperation from designers is an absolutely critical ingredient for a successful design.

3.1.3 It is difficult to validate a model when the design is still on the drawing board. There is nothing to validate against.

Simulation is the method of choice in designing new systems because of its flexibility and ease of understanding.

3.2 Tuning and Capacity Planning

In this case, the systems being modeled are well-defined. In most cases, the systems have been on the market for some time and documentation describing system behavior is provided by the system vendor. Systems on the market do not change as radically or as frequently as designs on the drawing board. In capacity planning, the modeler knows what is being modeled. The ability to measure running systems gives the modeler the opportunity to carry out intensive validation studies.

Analytic models can be used effectively for tuning and capacity planning studies. Validation against measured data should provide credibility for the model. Analytic models can produce output almost instantaneously, allowing the capacity planner to formulate and ask "what if" questions and get answers, interactively. Interactive dialogue with the model is very helpful, especially if the model produces graphic output and English-language-like textual output. Analytic models may be backed up by simulation models to enhance credibility and for very detailed analyses. In this mode, analytic models are used to evaluate a wide range of options rapidly, and simulation models are used for detailed study of a small set of points.

4. Critical Requirements for Modeling Packages

All good modeling packages must satisfy the following requirements:

- o A model should be shown to validate.
- o Ease of input. A model should be described succintly and naturally.
- o Useful input. A model should predict the performance metrics desired by the modeler.

In fine-tuning and capacity planning, most of the input to the model should come automatically from measurement and monitoring packages. In designing new systems, model input should come naturally from the descriptions of the designs. Since tuning and

capacity planning have been described at length, the remainder of this paper is focused on the user of simulation models in the design of new systems.

5. Simulation Languages

Most simulations are written in FORTRAN (or some other general purpose language) or GPSS. GPSS has the advantage of providing structures specifically for simulation, whereas these structures have to be explicitly encoded in FORTRAN. GPSS is a general purpose simulation system; it was not designed specifically for modeling computing systems. SIMPL/1 is another general purpose simulation language which is becoming popular since it provides simulation facilities and retains the capabilities of PL/1. In the last decade, languages designed specifically for simulation of computing systems have been developed. ASPOL, QSIM, and RESQ are pioneering examples. We next describe a system called PAWS which has the same design philosophy as QSIM and RESQ. The entire design of PAWS is based on one key observation: computer professionals prefer to describe their systems by drawing pictures. The ideal simulation system should take these pictures (or some representation of these pictures) as input. In practice, a modeler using a general purpose simulation system has to translate a graphical description of a computing system into a procedural description. This translation is often demanding of the modeler's time. The translation process is also the source of several errors. PAWS is designed to take a (computer readable) representation of a designer's picture as input. There is little translation required.

The most important question for developing a simulation modeling system ought to be: how do computer professionals describe computer systems to one another? We will address this question in this paragraph. Programmers and architects describe systems by drawing diagrams which are usually in the form of graphs. System resources are represented as nodes (vertices) in the graph. Edges from one node to another show the sequence in which transaction use resources. Usually, computer professionals use a hierarchy of diagrams to describe a system. At the top level of the hierarchy, a system is blocked out into large aggregates of resources; for instance, the entire I/O system may be represented by a single node in a graph. The model is refined at lower levels in the hierarchy. The description of the model refinement is itself another diagram. For instance, we may draw a diagram for an I/O system, where the nodes (resources) are channels, controllers, DASDs, etc., and the edges show the sequence in which these resources are acquired and released.

Figures 1 and 2 show a hierarchical series of diagrams describing a swap-based time-sharing system [1]. Figure 1 shows a network with two resources: terminals and a "computer". Figure 2 shows a refinement of the computer node, which represents an aggregate of system resources. The entry and exit ports of the computer node in Figure 1 are shown in Figure 2. Figure 2 shows that a computer really consists of CPU, I/O and MEMORY resources. Jobs entering the computer node acquire memory, then use the CPU and I/O a random number of times, release memory, and leave the computer node.

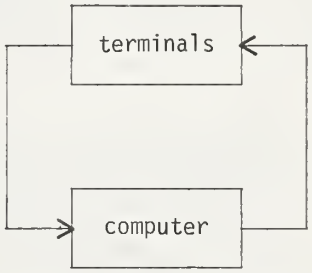


Figure 1. A Simple Model: Top Level of the Hierarchy

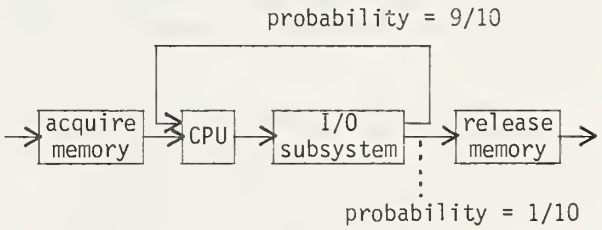


Figure 2. A Model of the Computer Node: Next Level of the Hierarchy

If we wished to, we could refine the description of the I/O subsystem further. The important point to note is that in a system engineer's mind, a model is a picture.

To analyze the model we need more information than the basic topology provides. Specifically, we need to know the scheduling disciplines by which the resources are managed, and the amount of each resource requested by each user. Normally, when a systems programmer says, "the CPU is managed according to a round robin fixed quantum

discipline with a 20 ms quantum," he expects to be understood. Unfortunately, most modeling systems do not "understand" terminology which is standard in the computing profession. So, to build a model of a computer system using a general purpose simulation system, the modeler has to translate the discipline, "round-robin-fixed-quantum," into a sequence of small operations such as "acquire processor," "hold for 20 ms," "release processor," etc. It is this detailed translation procedure which discourages people from using simulation models.

Memory is an important resource in computing systems, and most of the common memory management disciplines are described in the literature. A systems engineer is likely to describe a system by saying "system X has 512K words and uses a first-come-first-served, first-fit discipline." The statement is quite precise, but it may well require 300 lines of code in a general purpose simulation language. Generating this code takes time and may cause errors.

PAWS was designed to overcome these problems. PAWS has a large collection of routines to model most of the scheduling disciplines and resources one comes across in computing systems. For instance, PAWS "understands" the meaning of a "round-robin-fixed-quantum" discipline, and the meaning of 512K words of memory managed using a first-come-first-served, first-fit discipline." The input to a PAWS model is designed to represent the way in which one computer systems designer describes a system to another designer.

Resources modeled in PAWS include service resources (such as CPUs), memory, "passive" resources such as channels and controllers, and locks. Disciplines modeled in PAWS include first-come-first-served, preemptive priority, non-preemptive priority, round-robin-fixed-quantum, and polling. Memory management disciplines include first-fit, best-fit, and budy system. PAWS also has high-level language statements for generating different kinds of transactions, modeling parallelism (forks and joins), messages, routing, and processor interference (as in CPU-channel interference for memory). Users can also define their own types of resources (by writing FORTRAN programs) and then create PAWS models using these resources.

PAWS allows for succinct descriptions of complex computer systems. Users are supplied with documented PAWS source code which is also written in FORTRAN. PAWS is not a black box. Users may study the library of PAWS routines if they wish. Indeed, users are encouraged to add to the library and are provided with the facilities to do so.

6. Disadvantages of PAWS

PAWS was designed specifically for modeling computing systems, office automation systems, and information systems in general. However, many valuable constructs and scheduling disciplines in PAWS do not have application in modeling systems other than information systems (though its interface to user-written FORTRAN code allows a great deal of flexibility).

7. Summary

This paper is concerned with the question, "what are the characteristics of a modeling package which performance evaluation experts find most useful?" We showed that there are two distinct kinds of performance evaluation personnel: (a) those involved in designing conceptually new systems; and (b) those involved in tuning and capacity management. We gave reasons why: (a) simulation is the method of choice for those designing new systems; and (b) analytic techniques backed by simulation capability are the methods of choice in the tuning/capacity management situation. We emphasized that the most important characteristic of any modeling package is ease of user input. We observed that systems designers described systems to one another by drawing pictures. We described a simulation system (PAWS) which takes as input a representation of these pictures. The user of the system can add new modeling contructs and thus tailor the modeling system to the user's unique needs. We think that each performance evaluation group should adopt a convention for the pictorial representation of the systems that it evaluates. PAWS provides a convenient tool for mapping the pictorial representation into a running, accurate simulation.

References

[1] R. M. Brown, J. C. Browne, and K. M. Chandy., Memory Management and Response Time," CACM, Vol. 20, No. 3, pp. 153-165.

COMPUTATIONAL METHODS FOR QUEUEING NETWORK MODELS

Charles H. Sauer

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

A major objective of computing systems (including computer communication systems) development in the last two decades has been to promote sharing of system resources. Sharing of resources necessarily leads to contention, i.e., queueing, for resources. Contention and queueing for resources are typically quite difficult to quantify when estimating system performance. A major research topic in computing systems performance in the last two decades has been solution and application of queueing models. These models are usually networks of queues because of the interactions of system resources. For general discussion of queueing network models of computing systems, see C.H. Sauer and K.M. Chandy, *Computer System Performance Modeling,* (Prentice-Hall, 1981), and recent special issues of *Computing Surveys* (September 1978) and *Computer* (April 1980).

In the fifties and early sixties Jackson showed that a certain class of networks of queues has a *product form* solution in the sense that

$$P(n_1,...,n_M) = \frac{X_1(n_1)...X_M(n_M)}{G}$$

where $P(n_1,...,n_M)$ is the joint queue length distribution in a network with M queues, $X_m(n_m)$, $m = 1,...,M$, is a factor obtained from the marginal queue length distribution of queue m in isolation and G is a normalizing constant.

The existence of a product form solution makes tractable the solution of networks with large numbers of queues and/or large populations, but the computational methods are non-trivial unless the normalizing constant itself is easily obtained. This tutorial surveys the most important computational methods for product form queueing networks.

PRODUCTIVITY IN THE DEVELOPMENT FUNCTION

Phillip C. Howard

Applied Computer Research
Phoenix, AZ 85021

Productivity in the development organization is a function of several influences. Among these are influences which are strictly external to the development group, and over which it has little control, the collection of tools and techniques available to support development personnel, and the set of management policies and procedures which control the development function. The combination of these influences create a development environment which yields some level of productivity. This tutorial will address each of the key elements of this environment as depicted in Figure 1: the characteristics of the development organization, outside influences, tools and techniques, management control, and finally, productivity and its measurement.

To understand productivity in a development environment, it is necessary to understand something about the split between maintenance and new application development, the software development life cycle, and the allocation of effort to the various phases of the life cycle. Most studies have shown that an inordinate amount of effort is expended in the test and integration stage, which is directly related to errors and their causes. Various studies of errors, when and how they occur, and how they can be minimized are discussed. This ties in with the life cycle and the allocation of manpower to software projects. In addition, certain psychological aspects impacting the programming task are also considered.

Outside influences on the development organization are largely related to the nature of the interface with the user organization and the application itself. In particular, the experience of both data processing and user personnel in similar applications, and the availability of good requirements definitions (specifications) are critical to high levels of productivity. Another important factor has to do with the complexity of the application. Numerous studies have shown that there is a direct relationship between complexity and effort, and that productivity declines as complexity increases. Considerable attention is given to the question of complexity and how to keep it in control.

There are numerous tools and techniques available to development personnel, ranging

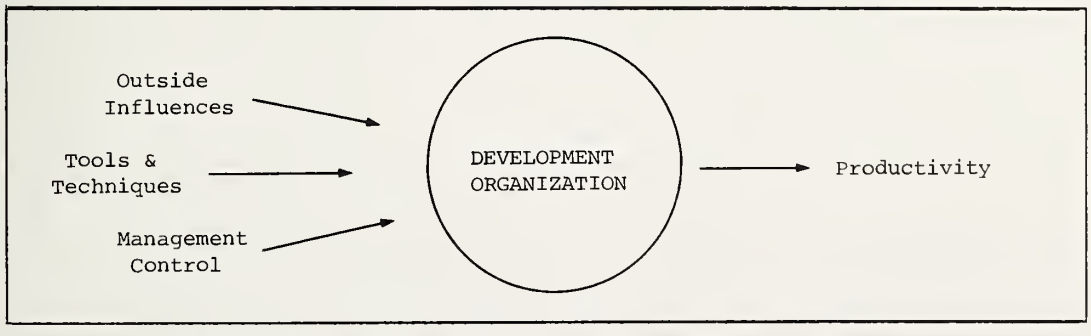


Figure 1. The Productivity Environment in Development

from design languages to debugging and documentation aids. Vendors of these products all claim significant productivity gains, but the potential compounding effect of several such tools never really seems to work out in practice. The various categories of these tools will be considered and some guidelines given on how to evaluate them in the context of a particular development environment. In addition, one of the special problems relating to the use of tools, that of properly integrating them into the working environment and overcoming people's natural inertia, or resistance to change, will be explored.

The specific management procedures that control the development organization also have an important influence on productivity. For example, project management not only allows for better control but entails a management discipline which contributes to improved productivity. The basic elements of software engineering have been shown to contribute to both improved software quality and improved productivity. Evidence is also shown to support the idea that structured programming and the general class of "improved programming techniques" can contribute to improved productivity as well. Problems of manpower allocation are discussed, including Brooks' "mythical man-month" and some ideas on how to allocate manpower to development work.

All of those influences on the development organization--external influences, tools and techniques, and management practices--determine the level of productivity actually achieved. First, productivity as a ratio of output to input is examined, with particular attention to the problems of defining or measuring "output' from a development organization. Two possibilities are considered, lines of code and "functions." Other types of quantitative measures are also discussed in the context of developing a productivity or performance "data base " for the development organization.

ACQUISITION BENCHMARKING

Dennis M. Conti

National Bureau of Standards
Washington, DC 20234

Abstract

Benchmarking has traditionally been more of an art than a science, primarily because of the previous lack of well-established procedures for constructing and testing benchmarks. As a tool for evaluating vendor performance during the competitive acquisition of computer systems, benchmarking has become an accepted practice. However, in spite of the widespread use of benchmarking, especially within the Federal Government, practitioners are still unaware of the sources of error that affect the ability of a benchmark to represent a real workload.

This tutorial will explore the purpose of benchmarking during the acquisition of computer systems, the sources of error inherent in the benchmark process, and ten steps to constructing and testing benchmark mixes. These steps have formed the basis for a recent NBS Federal Guideline in this area, FIPS PUB 75.

SOFTWARE CONVERSION PROCESS AND PROBLEMS

Thomas R. Buschbach

Federal Conversion Support Center
Office of Software Development
General Services Administration

This tutorial will provide users with a basic understanding of the software conversion process. The technical and management problems which arise during a conversion project will be discussed. A brief description of the Federal Conversion Support Center's capabilities to assist agencies in the conversion process will be given.

1. Federal Conversion Support Center

The Federal Conversion Support Center (FCSC) was established in May, 1980 as the Federal Government's primary source for software conversion technology. The intended effect of the FCSC is to ensure careful consideration of software conversion factors in ADP equipment and teleprocessing services acquisitions. The FCSC provides Federal Agencies with expertise, techniques and tools to conduct conversion studies and accomplish software conversions through the provision of reimbursable services.

2. Software Conversion Process

Conversion, as defined by the FCSC, is the transformation, without functional change, of computer programs or data to permit their use on a replacement or modified ADP system, telecommunications system, or teleprocessing service. Historically, Federal agencies have underestimated the resources required for software conversion due to a lack of understanding of the conversion process. A large part of the problem is the failure to realize all the tasks involved in a conversion. The conversion process consists of the following tasks: planning; materials preparation; translation; unit, system, and acceptance testing; maintenance changes; implementation. To accurately estimate the cost of a conversion the following cost elements must also be taken into consideraion; redocumentation, site preparation, training, management, and tools.

The conversion planning task includes; a complete inventory of current software, selecting tools to automate the process, defining standards for the converted programs and files, and a software conversion study which includes schedules and cost estimates. Some of the problems associated with the planning task include the large number of tasks and the amount of material to be handled.

The materials preparation task involves the generation and validation of test data, and the formation of conversion work packages. A system for tracking the work packages from the time they are formed until they are accepted back from the conversion contractor must also be developed. The problems associated with this task include the amount of test data required and its comprehensiveness.

The translation task encompasses the translation of all programs, files, JCL, and data bases. This would normally include a clean compile and testing at the program level.

The maintenance change task is required to track and eventually incorporate changes to the production software which occur during the conversion process.

The implementation task includes; system testing to ensure interoperability of programs, acceptance testing to validate output from the system, and cutover from the old to the new software in the production environment.

RESOURCE MEASUREMENT FACILITY (RMF) DATA: A REVIEW OF THE BASICS

J. William Mullen

The GHR Companies, Inc.
Destrehan, LA

Analysis and understanding of RMF measurement data is a basic prerequisite to performance management in the MVS environment. A basic approach to RMF analysis is described for key measurement data elements often overlooked in the analysis process, how RMF measures these values, and some rules-of-thumb based upon the author's experience. Specifics in the areas of I/O, paging/swapping, TSO, and analysis of cause and effect relationships of RMF measured data will be discussed.

Key words: Analysis; data elements; I/O; logical swapping; measurement; MVS; paging/swapping; TSO.

1. Introduction

The reduction and analysis of RMF data is a basic prerequisite to performance management and configuration tuning in the MVS environment. A considerable amount of material is available on the meanings of the measurement data reported by RMF and some documents attempt to provide basic 'rules-of-thumb' to aid the analyst in determining current performance levels for a configuration. A more basic approach to analysis of RMF data is the understanding of cause and effect relationships between the measurement data elements and the approach to determining interaction and subsequent courses of action to alleviate performance bottlenecks within the configuration.

Analysis of RMF data additionally requires a basic understanding of the System Resources Manager (SRM) and the data collection process whose data elements are subsequently reported by RMF. Algorithms used by the SRM to determine over and under utilization of the MVS system should be understood in general with constants in the algorithms that are applicable to the hardware in the configuration. The overall knowledge gained from study and understanding of the SRM algorithms provides a base for cause and effect analysis of RMF data. The SRM algorithms are included in Appendix A for reference.

Discussion of RMF measurement elements addressed in this paper will be done by RMF report type to provide ease of reference for measurement data elements.

2. CPU Activity Report

Primary measurement elements provided in this report are concerned with the processor average utilization during the specified measurement interval, the minimum, maximum, and average number of tasks for the three types of processing (batch, time-sharing, started task) and the minimum, maximum, and average values for the various queues within the MVS system.

A distribution of queue lengths is provided by RMF to give the analyst insight into the variances during the measurement period. Information for the CPU Activity Report is obtained by the RMF CPU sampling module MRBMFECP. This sampling module obtains the maximum, minimum, average, and distribution values by sampling the SRM User Control Block (OUCB) and the ASCB queues which are anchored in the Resource Manager Control Table (RMCT) addressed via the CVT. Counts and statistics gathered are stored in the ECPUDT data area.

A statistic of primary concern to the analyst is the 'Out-And-Ready' average value. In most instances, an average value greater than 1.0 for the measurement interval is usually interpreted as an indication of over-initiation of the system. The analyst should first investigate the data elements used by SRM in the over-utilization algorithm from all RMF reports (i.e. CPU activity, demand paging, UIC, etc.) to determine if the system is being operated in an over-initiated mode and there is a lack of needed system resources. If no symptom appears from this analysis, the analyst should then evaluate the Installation Performance Specifications (IPS) designations for minimum and maximum MPL of each Domain. This can be accomplished utilizing the RMF trace facility and specifying the DMDTCMPL and DMDTRUA trace key word values. The analyst should look for any domain that is consistently running at its specified maximum MPL. This could indicate an artificial constraint on the SRM's ability to raise the system target MPL and specifically the MPL for that particular domain. RMF trace values for the DMDTCMPL and DMDTRUA trace key words should be reviewed periodically to assure proper minimum and maximum MPL designations in the IPS.

An additional consideration is the 'Logical-Out-Ready' value and the effect on TSO logical swap. Logical swapping considerations will be addressed later in the paper.

3. Channel/Device Activity Reports

The MVS Operating System provides information statistics for physical and logical channels via the Input/Output Supervisor (IOS). The RMF channel sampling module, ERBMFECH, collects these statistics and provides information for both physical and logical channel activity.

Primary interest in the physical channel reports revolves around the average service time for the channel. This value is the calculated average time required for the channel to complete an I/O operation. Averages for channel I/O operations tend to run in the 10 to 15 millisecond range. As this value increases, a significant change may be seen in the channel busy/CPU wait percentage though the change may not be rapid enough to attract attention. A more significant indicator of channel and I/O problems are the 'Average-Que-Length' and '% Request Deferred' fields on the logical channel report. MVS associates each physical channel and each primary/optional channel specification with a logical channel. Of primary interest in the logical channel report is the 'Average-Que-Length' field. This field represents the depth of I/O requests queued for the logical channel. A value of .1 or greater for this field is an indication of possible channel overload. Though a direct correlation may be seen between this field and percentage of channel utilization, channel utilization is generally not the main contributor to the problem.

The percentage of request deferred distribution provides the major indication of the problem. If requests deferred due to channel busy is the high percentage, the device activity report should be consulted to determine the volume or volumes receiving the highest I/O activity. This will give a starting point to determine if a volume move or data set moves can be done to alleviate the problem.

If the percentage of request deferred due to control unit busy is the high percentage, then three alternatives must be investigated. The first is rerouting of channel through alternate or additional control units. This is usually not a viable alternative. The second option is the movement of DASD strings to other less utilized control units. This can be a very viable alternative in concert

with proper DASD segregation by workload at the DASD string level. This effort will revolve around utilizing the device activity reports to again identify the volumes receiving the most activity and making appropriate volume or data set moves to distribute I/O activity. The final alternative is necessary if channel paths from more than one processor are routed through the control unit pairs. Workloads on each processor must be evaluated with respect to their use of the DASD devices on the strings serviced by the control units to determine if movement or combining workload processing between processors will allow a better I/O distribution to the channel paths being serviced by the control units. A final consideration is the channel algorithm (i.e. rotate, LCU) being used on each of the processors. With proper DASD segregation by workload at the string level, the Last Channel Used (LCU) algorithm has proved best for most installations (assuming an MVS/SE1 or higher environment). Processors will tend to sort channel and control unit paths out between themselves which moves DASD contention down to the string, and more specifically, the device level.

When device busy is the highest percentage for deferred requests, the device usage must be studied to determine which data sets are responsible for the high activity level and consider either a consolidation of workloads to a given processor or duplexing the data set (s) involved. Device busy contention will only occur in a multi-processor environment.

The objective of analysis of RMF channel/device activity is the identification of I/O overload conditions and more importantly, the elimination of contention in the multi-processor environment. DASD contention at the control unit level is most difficult to resolve. Through workload segregation and use or proper channel algorithms, the analyst should attempt to transfer any contention to the string and device level. Once this transfer is accomplished, the contention can be alleviated through volume and data set moves on existing DASD strings.

4. Page/Swap Data Set Usage

Resource Measurement Facility collects information on page data sets, swap data sets, and system paging activity via the paging sampling module ERBMFEPG. The sampling module extracts the data from two sources: the Paging Vector Table (PVT) and the Auxillary Storage Manager Vector Table (ASMVT).

Analysis of the Page Data Set Usage report and the Swap Data Set Usage report should be directed toward the following elements:

- bad slot/swap set indications for page or swap data sets;
- average service times for allocated page and swap data sets; and
- allocated page/swap data set slots and swap sets with minimum, maximum, and average slots/swap set indicators.

An increase in page data set service times is usually caused by an increase in cross system DASD contention. Another cause of increases in service times for local page data sets can be a spill of swap activity, due to a shortage of swap sets and swap data sets, into the local page data sets. Increases in the number of logged-on time-sharing (TSO) users is usually the cause of swap data set spills. The analyst should continuously review the statistics on swap set usage. If the maximum swap sets used is equal to swap sets allocated for all in-use swap data sets and is a frequent condition, swap spill to local data sets is probably occurring. This will also be in concert with an increase in average ended transaction times for TSO transactions.

Bad slot/swap set counts for page or swap data sets can indicate the beginning of DASD volume problems and induce intermittent abnormal task terminations into the system. This is particularly true if a bad slot indicator occurs for the PLPA page data sets. If bad slot indications go unnoticed for the PLPA page data set, the potential for a spill of PLPA into the common page data set is present with system performance degradation

being the end result.

An additional consideration is the size and placement of page data sets. Analysis of the usage of the page data sets from the usage report will provide the analyst information on page slots used with respect to page slots allocated. Several page (local) data sets should be used and should be allocated with a minimum of slots over the measured maximum slots used. This reduces the seek activity due to the spread of pages over slots by the ASM slot sort routine.

A final point is the use of the RMF trace facility to track page delay time. The trace key word, RCVMSPP, provides the average page delay time (excluding swap pages) in milliseconds for transfer of pages during the measurement interval. This value will alert the analyst to paging problems as an initial indicator and tends to be less subtle than other indicators discussed.

5. Paging Activity Report

Paging can be the absolute controlling factor in the performance of an MVS system if the analyst does not recognize the options available to control the system paging rate. Two values derived from the Paging Activity Report are key in the SRM's system control decision making process:

- Demand paging - number of non-VIO, non-swap page-ins plus page-outs per second; and
- Page fault rate - number of non-VIO, non-swap page-ins plus page-reclaims per second.

The analyst should track these values to determine if they are approximately equal. In most MVS installations, page-ins are the primary contributor to each of these values. The third parameter required is the page unreferenced-interval-count (UIC) average during the interval. This value must be obtained through the RMF trace facility by use of the RCVUICA trace key word. The average UIC value is a primary value used by the SRM to determine system over-under utilization of the system and tends to be the most erratic of the three variables where systems are not memory (real) overcapacitated. The demand paging value is used in concert with other variables in the SRM's decision making process with only UIC and page-fault rate being the stand alone variables in the algorithm. The analyst should evaluate the feasibility of changing the SRM constant values to negate the UIC value as a control and utilize the page-fault rate (PTR) to control system paging activity. This can be accomplished by altering the SRM constants located in CSECTIRARMCNS in the nucleus for MVS/SE Release 1 or prior systems and in SYS1.PARMLIB member IEAOPTXX for MVS/SE Release 2 and MVS/System Product (SP) systems.

The second portion of the Paging Activity report displays statistics on system swapping activity during the measurement interval. Logical swap counts and their relationships will be discussed under the section on TSO logical swapping. Swap sequence counts provide the total swapping activity count for the interval and averages for pages swapped in and out during the interval. The primary element here is the swaps per second count. In a system with sufficient swap data sets allocated, the analyst should try to maintain a 1.0 swap per second or less average. If this value tends to be greater, the physical swap counts section should be analyzed.

Within the physical swap counts section, a primary element often overlooked is the 'detected wait' swap count field. Many analyst simply write this count off to system initiators going inactive. The analyst should investigate the minimum, maximum, and average values for active initiators from the CPU Activity Report and determine if this is actually the cause. If not, the other major contributor to this parameter is DASD wait time. If investigation of paging activity and paging service times appears to be within reasonable bounds, the analyst should begin investigation DASD path and device queueing. Normally this investigation will show that a given path or device (s) on the path are experiencing significant I/O activity, cross system contention, and queueing of I/O request. The analyst can use SMF records to determine the type of workload requesting I/O service to the

DASD in question, from each processor, and begin attempts to provide better workload segragation at both the processor and DASD level.

Swap counts for unilateral and exchange on recommendation value swap counts are associated with IPS design and parameters and are not in the scope of this paper.

Additional swap reason counts are self explanatory to the analyst. Determining the workloads affected by the above swap reasons can be accomplished through use of the RMF background monitoring facility and reduction of RMF type 79 SMF records.

6. TSO Logical Swapping

The TSO logical swap or in memory swap was introduced with the release of the MVS/Systems Extensions (SU50). At swap out time (terminal input or output wait), the TSO user is evaluated for logical swap. If the criteria for logical swap is met, the TSO user's fixed frames, recently referenced working set frames and LSQA remain in storage and the user is placed on the out/wait queue to await terminal input. If terminal input is not received during a specified period (think time), the TSO user is physically swapped out of memory. If terminal input is received within the specified amount of time, the TSO user is moved to the out/ready queue to compete with other tasks waiting to be swapped in under normal SRM control.

The control variable for TSO logical swapping is a value known as 'think time' and is the amount of time the user is allowed, after being logically swapped out, to enter data at his terminal and prevent being physically swapped out. The maximum default think time is 30 seconds and is in the SRM constants table (IRARMCNS). An additional value used in the logical swap decision is the system average available frame count (RCVAFQA). Though the ASM queue length is also part of the algorithm for the logical swap determination, it is negated due to default values assigned which should not be changed by the analyst.

Primary controls for raising or lowering the system think time are the system average UIC (RCVUICA) and the system average available frame count (RCVAFQA). If the system average UIC is greater than 30 and the system available frame count is greater than 300, the system average think time is raised .5 seconds. If the system average UIC is less than 20, then the system average think time is reduced 1 second regardless of the system average available frame count.

Three problems are present with the algorithm and the current default values:

- o the system UIC can be somewhat erratic depending upon the workload mix processing on the system;
- o the UIC comparison values of 20 and 30 are high; and
- o the available frame count value of 300 is too high.

The analyst should review the logical swap statistics from the paging activity report to determine the degree of logical swapping in the system. In concert, the average available frame count should be reviewed to determine if the average available frames is consistently greater than 100. If this is the case, the analyst should consider reducing the default compare value of 300 to 100 and evaluating the results for logical swap.

After taking the above action and reviewing results, the analyst should trace the RCVUICA value for minimum, maximum and average values to determine UIC ranges under a represenative workload. If traces show the UIC value to be in a range less than 30 but greater than 10, the analyst should consider reducing the default values of 20 and 30 for the UIC compare and again review the effects on logical swap.

Tracing the system average think time should be done in concert with the above actions. If the system think time remains high or at the maximum value without an increase in the percentage of TSO users being logically swapped, the analyst must assume that the user community has a large average

think time for the terminals in use. The analyst may want to consider raising the default maximum system think time to induce more logical swapping into the system. This should be done only if sufficient memory is available and the increases in maximum think time are gradual with a thorough review on the effect to logical swap.

The objective of increasing the percentages of TSO users logically swapped out and subsequently swapped back in is to reduce the instruction path length and the channel/device overhead associated with physical swapping. Care must be taken to avoid inducing additional demand paging and memory shortage overhead into the system with TSO logical swap improvements.

7. Summary

Analysis of RMF data requires an understanding of the cause and effect relationships of those variables used by the SRM to control the MVS system. Direction for understanding these relationships and analysis of the data can be gained through the following:

- o review and understanding of the SRM algorithms for over and under utilization of the system;
- o review and understanding the TSO logical swap algorithm;
- o review and understanding of the cause and effect relationships discussed in this paper; and
- o utilization of the RMF trace facility to obtain the key values used in the SRM algorithms and analysis of the relationships in your system.

Exercising the above recommendations will give the analyst a substantial start in the RMF data analysis process.

8. Appendix A.

The SRM algorithms and TSO logical swap algorithms are included for reference.

```
Over utilization(SRM):

    ( UICA <   2 ) or
    ( CPUA > 100 ) or
    ( ASMQ > 100 ) or
    ( PTR  > 100 ) or
    ( MSPP >1000 ) or

 ( DPR >DPRTH ) and (( CPUA >98% or
                       ( MSPP >130))

    or

 (( IOUSE = '1'B ) and ( CPUA < 98% ))

 ( TMPL > MINMPL ) for at least 1

 Domain.

TSO Logical Swap(think time increase);

 IF ( UICA  > LSCTUCTH   (30) and
    ( AMSQA < LACTASTL  (100) or
    ( AFQA  > LSCTAFQH  (300) then

   LSCTMTES = MIN(LSCTMTEH,LSCTMTES
                  + LSCTMTEI )

TSO Logical Swap(think time decrease);

 IF ( UICA    LSCTUCTL < (20) or
    ( ASMQA   LSCTASTH >(100) or
    ( AFQA    LSCTAFQL <  (0) then

   LSCTMTES = MAX(LSCTMTEL,LSCTMTES
                  - LSCTMTED )
```

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. SP 500-83	2. Performing Organ. Report No.	3. Publication Date November 1981

4. TITLE AND SUBTITLE

Computer Science and Technology: Proceedings of the Computer Performance Evaluation Users Group, 17th Meeting. "Increasing Organizational Productivity."

5. AUTHOR(S)

Terry W. Potter, Editor

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as no. 6

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 81-600155

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

These Proceedings record the papers that were presented at the Seventeenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 81) held November 16-19, 1981, in San Antonio, TX. With the theme, "Increasing Organizational Productivity," CPEUG 81 reflects the critical role of information services in the productivity and survival of today's organization, as well as such trends as increasing personnel costs, limited budgets, and the convergence of data processing, communications, and word processing technologies. The program was divided into three parallel sessions and included technical papers on previously unpublished works, case studies, tutorials, and panels. Technical papers are presented in the Proceedings in their entirety.

12. KEY WORDS: Benchmarking; capacity planning; chargeback systems; computer performance management; data base machines; end user productivity; human factors evaluation; information system management; office automation; performance management systems; resource measurement facility; simulation; supercomputers.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D C 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

320

15. Price

$9.00

USCOMM-DC 6043-P80

☆U.S. GOVERNMENT PRINTING OFFICE: 1981-360-997/1850

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $16; foreign $20. Single copy, $3.75 domestic; $4.70 foreign.

NOTE: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

DIMENSIONS/NBS—This monthly magazine is published to inform scientists, engineers, business and industry leaders, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, pollution abatement, health and safety, and consumer product performance. In addition, it reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing. Annual subscription: domestic $11; foreign $13.75.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).

NOTE: The principal publication outlet for the foregoing data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Services, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services, Springfield, VA 22161, in paper copy or microfiche form.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

SPECIAL FOURTH-CLASS RATE
BOOK