

AD-A104 888

BATTELLE COLUMBUS LABS OH

F/G 9/2

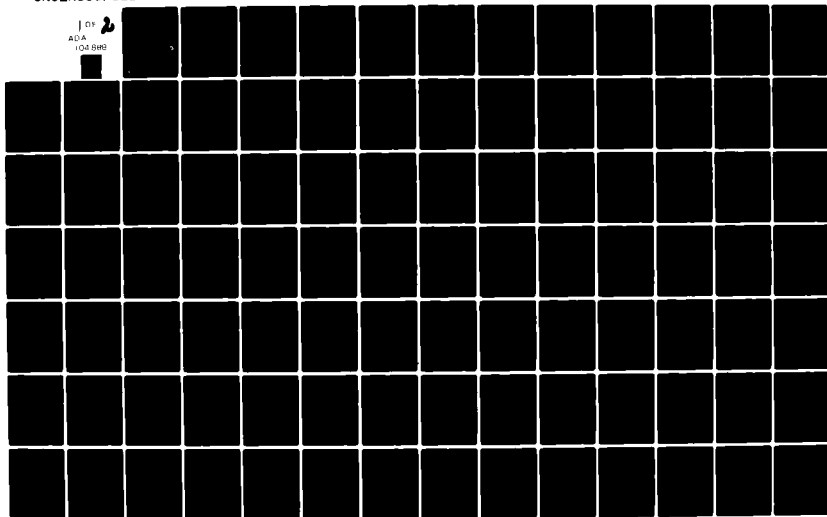
AN IPSS-BASED MODEL-BUILDING METHODOLOGY FOR RANKING AND EVALUA--ETC(U)

OCT 79 J D BROWNSMITH, J S CARSON

NL

UNCLASSIFIED

JDS
ADA
104 888



LEVEL

AIRMICS FINAL REPORT*

**An IPSS-Based Model-Building Methodology
for Ranking and Evaluating Computer
Hardware/Software Systems**

(15)

AD A1 U4888

By

Joseph D. Brownsmith, Ph.D.
Database Systems Research and Development Center
Department of Computer and Information Sciences
University of Florida
Gainesville, Florida 32611

John S. Carson II, Ph.D.
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

William H. Hochstettler III, M.S.
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

15 October 1979

DTIC
SELECTED
OCT 1 1981
H

* This research was performed for AIRMICS through the United States
Army Research Office with the financial support of the Battelle
Scientific Services Program-D.O. 1271

DISTRIBUTION STATEMENT
Approved for public release
Distribution Unlimited

DTIC FILE COPY

81 10 1 055

AIRMICS FINAL REPORT

6 An IPSS-Based Model-Building Methodology
for Ranking and Evaluating Computer
Hardware/Software Systems,

By

1 Joseph D. Brownsmith, ~~John S. Carson II~~
Database Systems Research and Development Center
Department of Computer and Information Sciences
University of Florida
Gainesville, Florida 32611

John S. Carson II, Ph.D.
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

William H. Hochstettler III, M.S.
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

11 15 Oct 1979

* This research was performed for AIRMICS through the United States
Army Research Office with the financial support of the Battelle
Scientific Services Program - D.O. 1271.

The views, opinions, and/or findings contained in this report are
those of the authors and should not be construed as an official
Department of the Army position, policy, or decision, unless so
designated by other documentation.

Executive Summary

The objective of this research was to design and implement a model building methodology for simulating U.S. Army computer hardware/software systems. Computer systems are characterized in terms of file parameters, hardware specification, and software use of files. These descriptions reside in a model library and are the building blocks in the model synthesis process. The Information Processing System Simulator (IPSS) language was used to encode these descriptions and to represent the sequence of computer activities for application program processing (e.g., job scheduling, buffer management, channel program).

Two computer systems were compared using this methodology. Simulation models were written for an IBM 360 Model 30 computer and a Honeywell Level 6 minicomputer. A subset of the U.S. Army Standard Installation/Division Personnel System (SIDPERS) provided a common loading for both systems. Data was collected on an operational IBM 360/30 and the IPSS model was validated. The statistical results, derived from IPSS, indicate resource utilization (for both hardware and software resources), elapsed time, and queueing. Our results project that an eight hour execution of SIDPERS on the IBM 360/30 would execute in approximately two and one-half hours on the Honeywell machine. Models of several hardware variations were prepared in order to demonstrate responsiveness capabilities of the methodology. A manpower analysis is provided for guidance in estimating future work.

Accession for	
Dist	
BY	
Distribution of	
Availability of	
Special	

40 or file

ACKNOWLEDGMENTS

The Authors would like to acknowledge the able project administration and support of Mr. James Gantt and Mr. Allan Curry of AIRMICS. We also thank CPT Paul A. White and 1LT Robert H. Fleming of Quality Assurance, USACSC, for providing data and other information, and for making valuable suggestions on the content and presentation of this final report. In addition, MAJ Schuler, SP5 Sweatman, and SGT Huerd of the Fort Stewart Division Data Center provided us with much valuable operating data and SIDPERS data, and allowed us to observe a SIDPERS run. Mr. Dixie Tompkins, of the Management Information Systems Office, FORSCOM, Fort MacPherson, also provided us with data, manuals and other valuable information about SIDPERS software and files. Finally, we deeply appreciate the work and dedication of Mr. Bruce E. Wilhite, an undergraduate student at the University of Florida, who installed and maintained IPSS and provided computer support during the project.

TABLE OF CONTENTS

	Page
EXECUTIVE SUMMARY	i
ACKNOWLEDGMENTS	ii
LIST OF TABLES	v
LIST OF FIGURES	vi
1. INTRODUCTION	1
1.1 Summary of Research Activities	
1.2 Summary of Results	
2. BACKGROUND	6
2.1 Background to the Project	
2.2 Approach to Modeling and Validation	
3. THE IPSS APPLICATION PROCESSING SYSTEM METHODOLOGY (IAPS)	12
3.1 The IAPS Modeling Perspective	
3.2 The IAPS Modeling Approach: The User's Role	
3.3 The IAPS Modeler View of Computer Hardware/ Software Systems	
3.4 The IAPS Simulator View	
4. MODELING SIDPERS USING IAPS	36
4.1 Selection of a SIDPERS Subset	
4.2 Modeling the SIDPERS Subset	
5. MODELING TWO COMPUTER HARDWARE ARCHITECTURES USING IAPS	44
5.1 IPSS Hardware Characterization	
5.2 Architectural Variations	
6. OVERVIEW OF IAPS MODEL STRUCTURE AND EXECUTION . . .	55
7. ANALYSIS OF RESULTS OF THE IAPS SIMULATIONS	62
8. SUMMARY OF PROJECT ACTIVITIES	74

Table of Contents continued.

	Page
9. SUMMARY AND CONCLUSIONS	80
LIST OF REFERENCES	89
APPENDICES	91
A. An Overview of the Informaton Processing System Simulator (IPSS)	
B. IAPS Source Code	
C. SIDPERS P1A Through P1G Processing Characteristics	
D. Examples of IAPS Input Tables	
E. The Model Library	
F. Guide to Preparing IAPS Input	
G. GRASP Accounting Data	

LIST OF TABLES

<u>Table</u>	<u>Page</u>
3-1. Modeling Activities	17
3-2. Modeler Checklist for Computer System Hardware . .	25
3-3. IPSS Data Required to Model I/O Devices	26
4-1. Sources of System Data	39
4-2. 2 August 1979 Ft. Stewart Record Processing for Programs PlA Through PlG	40
4-3. SIDPERS Basic Cycle, 2 August 1979, Ft. Stewart (Extracted from GRASP Accounting Reports)	43
5-1. Direct Access Storage Devices	47
5-2. Magnetic Tape Units	48
5-3. Hardware Differences	51
5-4. Performance Characteristics of Alternate Hardware Architectures Relative to Model A1 (Standard 360 Model 30)	54
6-1. Index to Modeled SIDPERS Processes	59
7-1. Validation Results	65
7-2. Simulation Elapsed Time Per Job Step (minutes: seconds)	67
7-3. System Comparison	68
7-4. Simulation Elapsed Time Per Job Step	72
8-1. Breakdown of Project Activities	75
8-2. Manpower Analysis and Projection	77
C-1. File Characteristics for Programs PlA Through PlG .	137
G-1. GRASP Accounting Data	161

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
3-1. Overview of the IAPS Methodology	14
3-2. User's Role in the IAPS Methodology	18
3-3. User's View of IAPS Model Synthesis and Execution .	20
3-4. Example of Interactive Model Synthesis	21
3-5. User's View of IAPS - Proposed	22
3-6. Modeler's Role in the IAPS Methodology	23
3-7. An Example of the Application File Table	28
3-8. An Example of the Application Processing Table . . .	29
3-9. An Example of the System File Table	32
3-10 Overview of the Service Hierarchy in the IPSS Model - The Simulator's View of the IAPS Methodology	33
3-11 IPSS Program Control and Data Interfaces - The IPSS Analyst View of IPSS (DEL78a).	35
5-1. Typical CS ₃ Hardware Configuration	45
5-2. Honeywell Level 6 Architecture - DAS ₃	46
6-1. Simulator Overview of the IAPS Service Hierarchy . .	57
6-2. Summary of Models Synthesized for Evaluation of Alternate Hardware Configurations	58
A-1. The IPSS Methodology	93
B-1. An Example of IAPS Source Code	105
C-1. PLA File Identification	133
C-2. PLB File Identification	134
C-3. PLC File Identification	135
C-4. PLG File Identification	136

List of Figures Continued.

<u>Figure</u>	<u>Page</u>
D-1. Explanation of Headings in Figures D-2 and D-3 . . .	140
D-2. Application File Table for SIDPERS	141
D-3. System File Table for SIDPERS	142
D-4. Application Processing Table for SIDPERS (Jobsteps PlA and PlB)	143

PRECEDING PAGE BLANK-NOT FILMED

1. INTRODUCTION

This report is in response to the Laboratory Research Cooperative Program Statement of Work TCN: 79-245, which required the services of three research scientists on a short-term project to develop simulation models of computer systems. The objective of this research was to produce a model building methodology using the Information Processing System Simulator (IPSS) to develop a ranking and evaluation procedure for computer hardware/software systems. Five specific tasks were identified:

1. Using IPSS, specify, design, build, test, validate, verify and document a model of an existing Army computer hardware/software system (such as the U.S. Army Base Operations System (BASOPS) implemented on IBM 360/40 equipment).
2. Using IPSS, specify, design, build, test, validate, verify and document a model of an advanced Army computer hardware/software system (such as a minicomputer data base oriented system).
3. Specify and collect data needed to build the models of computer hardware/software systems specified in 1 and 2 above.
4. Develop measures to allow for the ranking and evaluation of computer hardware/software systems. Factors should include, but not be limited to, growth rate, workload, software, variance in configurations, and new applications.
5. Arrange for and provide computer support services, to include computer time, disk storage, and IPSS software support.

This final report to AIRMICS reflects the background activity, purpose, procedures, documentation, and summary of work performed under each of the above tasks.

Tasks 1, 2, 3 and 5 have been completed in full; we could not complete task 4 due to lack of time. As the Army is considering replacement of certain of its computer systems, we did consider, relative to task 4, measures for evaluating computer systems when the major factor is variance in hardware configurations.

1.1 SUMMARY OF RESEARCH ACTIVITIES

The primary objective of this project was accomplished by designing, building, testing, verifying, and validating two basic models of Army computer systems. The first model was of an existing Army computer system, namely, an IBM 360 Model 30 with a selected subset of the Standard Installation Division Personnel System (SIDPERS) basic cycle for loading. This model provided a frame of reference and was validated. The second model was of an advanced computer system (one not currently operational) that was considered to be typical of potential Army purchases. This system was a Honeywell Series 60 Level 6 Model 47 minicomputer with the same SIDPERS basic cycle for loading. Several variations on the basic hardware architecture were modeled and analyzed.

These models were compared against the same workload, the first four jobsteps of SIDPERS. The results of such comparisons allow for a relative ranking of the various systems. Such a ranking is the first step towards determining what computer will meet the needs of the location

being examined. The IPSS approach is unique in that almost all currently available simulation techniques deal only with representative batch oriented systems while IPSS has special facilities which will allow the modeling of advanced computer features such as data bases, networks, and interactivity.

Of primary concern is the acceptance of the modeling methodology within the Army. Thus we have concentrated on validating a model of a simple and typical hardware/software system. The underlying assumption is that credibility will transfer to models of more advanced systems given a well validated basic model.

At the start of the project, AIRMICS personnel provided us assistance in selecting the computer systems that we were to model. After a preliminary investigation of the type of processing SIDPERS performs and the hardware configurations, we proceeded as follows:

1. Developed the IPSS Application Processing System (IAPS) methodology for representing application systems processing.
2. Implemented the methodology in IPSS.
3. Collected data on four SIDPERS job steps and the two computer hardware configurations.
4. Coded the hardware and software descriptions for the above in IPSS.
5. Verified the IPSS models.
6. Validated the model of the IBM 360/30.
7. Performed experiments using alternate hardware configurations.
8. Analyzed the results.

1.2 SUMMARY OF RESULTS

This section summarizes the major contribution of our research project. These results are discussed in detail in the sections referenced.

First, we demonstrated the appropriateness of using IPSS for modeling typical U.S. Army computer hardware/software systems, and showed that the simulation technique can provide data useful for the comparison of alternative computer systems. To ease the task of modeling in IPSS, we provided a high level modeling approach through our IPSS Application Processing System (IAPS) methodology which is able to accommodate any level of detail desired by the simulation user. (See Chapter 3)

In conjunction with IAPS, we established a basic library of model components which allows a user to easily and quickly build a model of a large number of design alternatives. This library can be modified and the number of its members increased so as to enhance future Army modeling needs. The library as it currently exists is described in Appendix E.

We identified the types of verification and validation data needed and their sources within the U.S. Army Computer Systems Command. (See Chapter 4) The ready availability of needed data would greatly shorten the time needed to complete any future modeling efforts.

We demonstrated the feasibility of the IAPS methodology, and the usefulness of the data collected by modeling and comparing several design alternatives for the Army CS₃ hardware/software system. This

effort included validation of a model of an existing system, development of models of nine alternative hardware configurations, and a comparison of the different systems. (See Section 5.2 for the hardware alternatives, Sections 4 and 5 for the SIDPERS model, and Section 7 for the simulation results.) In addition, we have given a manpower analysis of the current project and several possible future projects. (See Section 8.)

We identified appropriate measures for the comparison and ranking of production, batch oriented Army computer systems. Those measures which can be estimated via simulation, and which are collected automatically by IPSS, include job elapsed time, resource utilization, and queuing statistics. (See Section 7.)

All in all, we believe that the complete IAPS simulation methodology is a feasible and potentially useful approach in the Army's evaluation and comparison of alternative computer systems.

The remainder of this report is organized as follows. In Section 2, we discuss the background to the project and the motivation for our modeling approach. The methodology for modeling and evaluating computer hardware/software systems is presented in Section 3. Sections 2, 4, and 5 present the specific problems of modeling SIDPERS and the selected computer hardware architectures. Section 6 identifies the structure of our IPSS model while Section 7 presents the results of our modeling experiments. A summary of the time required for various modeling activities is given in Section 8. We finish with a summary, recommendations and conclusions in Section 9. A number of Appendices contain auxiliary material.

2. BACKGROUND AND APPROACH TO COMPUTER EVALUATION

2.1 BACKGROUND TO THE PROJECT

The United States Army is about to enter a period in which several large purchases of computer hardware/software systems are to be made. For example, one purchase could involve the replacement of over 40 computer installations. Choosing one machine to work at over 40 places would be complex enough, but in this case theoretically there could be over 40 different machines chosen. A computer vendor can bid on any number of sites with any combination of equipment. The workload profile at the locations for the machines is radically different. A minicomputer might work very well at one place while another must have a large main-frame.

Currently the sites have IBM 360/30's, IBM 360/40's, and IBM 360/50's, some with single peripherals, some with dual peripherals, some running the DOS operating system (or the enhanced DOS system DOS-E) and some with OS. A major portion of the software is consistent from location to location, but the volume of transactions is drastically different. All current work is batch oriented.

The Army desires to purchase new equipment to replace these machines. They want to add interactive capabilities while retaining batch processing for some applications. One requirement of the new computers is that they process the current work in one eight hour shift five days a week. (Currently most sites are running 24 hours a day five to seven days a week).

With recent technological advances, selecting even one computer is almost beyond human capability if one is to easily and fairly compare all of the machines that vendors would contend can do a given job. Current methods, such as benchmarking, fall short of solving the problem. Simulation appears to be a very attractive approach because of its flexibility and power in representing complex activities. Thus, this project requires the use of discrete event digital simulation to assist in the selection and evaluation of computer hardware/software systems.

This project specifically required the use of the Information Processing System Simulator (IPSS) to rank and evaluate computer hardware/software systems. IPSS is a special-purpose discrete event digital simulator system which was specifically designed to facilitate the investigation of the behavior of complex computer-based information processing systems (DEL77, DEL78a).

One significant feature of IPSS is its ability to characterize a computer's I/O subsystem. The IPSS language contains a rich set of instructions for describing control units, channels, disk and tape drives, and unit record equipment. The IPSS "service" concept permits a flexible characterization of the acquisition, use and release of the secondary storage "facilities".

These features are important because of the Army's predominately I/O oriented computer systems. Thus, a detailed modeling of the I/O subsystem should be of great potential value in identifying bottlenecks and effects of hardware/software changes. IPSS provides the capacity for detailed modeling of this computer subsystem and

also automatically collects elapsed time, resource utilization and queueing statistics for the user.

Although IPSS is a prototype system, it proved to be an able tool for characterizing salient features of SIDPERS application as well as the hardware characteristics of the IBM Model 30 and the Honeywell Level 6 minicomputer. An overview of the IPSS methodology is provided in Appendix A.

2.2 APPROACH TO MODELING AND VALIDATION

The completion of the specific research tasks outlined in Section 1 required the resolution of two basic issues, namely:

1. How to represent application program software in a simulation model, and
2. What data was available within the U.S. Army for validation of simulation models of computer hardware/software systems.

Since our resolution of these tasks was both time consuming and crucial to the results of the project, an overview of our approach is given here.

The first task, how to model computer software, can be rephrased as: What is the best approach for modeling large software systems by using IPSS. (IPSS has considerable flexibility and power in representing computer hardware, so that aspect of computer system modeling was not a problem). Software systems such as SIDPERS contain many large COBOL programs. We recognized that a detailed statement-by-statement or even paragraph-by-paragraph description of the processing logic

would be far too time consuming for this project. IPSS is, however, capable of representing processing logic at this level of detail and this approach may prove to be valuable for some programs, such as operating system routines, or for more refined results. Even if this approach were used, only one or two simple programs could be characterized in the time allowed, giving little insight into the processing characteristics of a large system. Clearly, we were challenged to produce a faster modeling approach which would both retain a useful level of accuracy of the detailed approach and provide flexibility for the modeler.

SIDPERS, the software system selected, has been modeled before using the software simulation package CASE (ADL75, SWE76). In addition, an IPSS model of several SIDPERS programs was part of an IPSS SIDPERS/IDMS simulation study (BRO77, DEL78), and a DIMUI model also simulated these programs (SCH77). The methods for representing software processing in these models were studied but not adopted in our methodology. The CASE approach represents files and file processing in an easily-understood and consistent manner, but overall was not judged to be a suitable methodology because of its lack of flexibility (for an evaluation of CASE versus IPSS see ROS78). The previous IPSS and DIMUI approaches were rejected since they modeled interactive SIDPERS programs by representing every call to a DBMS routine. (The batch SIDPERS programs that we modeled requested I/O to many types of files in the absence of a DBMS).

Our approach was to detail I/O processing on a file by file basis, and, in less detail, to characterize program CPU loading. This is consistent with the IPSS methodology and also allows the

modeler freedom to change the procedural structure of the model. Our methodology, called IAPS (IPSS Application Processing System), is reported in Section 3.

The second task we faced was the selection of a hardware/software system for which validation data existed. Validation is the process of determining the degree of validity of a simulation model. A valid model is one which is capable of accurately measuring, predicting and representing a system. The validation process proceeds to build an acceptable level of confidence that an inference about a simulated process is a correct or valid inference for the actual process. Validation of a model is performed by a comparison of the recorded observations of the real process with simulator outputs from a verified model, thereby establishing the versimilitude of the model and the real world process (MIH76a, MIH76b). Seldom, if ever, will validation result in a "proof" that the model is a correct or "true" representation of the real process (VAN69). Verification, on the other hand, is the comparison of the model's responses with those anticipated if the model's structure were programmed as intended (MIH76b). This means testing the outputs of the random number generators as well as checking that the computer program correctly executes the logic desired by the modeler.

With assistance of USACSC personnel we decided, early in the project, to model a subset of SIDPERS executing on the IBM Model 30 utilized at the Division of the Army's organization and to compare the results against a Honeywell Level 6 Model 47 minicomputer using the same SIDPERS workload. We also determined that GRASP step accounting data was the most important requirement for a detailed validation of our simulation models. Detailed validation of a SIDPERS job step requires the following:

1. GRASP step accounting data
2. The operator console's log (for the job step)
3. SYSLIST (for the job step)
4. Listing of specified data sets
5. VTOC listing of all disk packs which were on-line during the job step
6. Researchers present in the machine room during execution of the job step.

Since GRASP step accounting data was not available on a 360/30 but was available on a 360/50, we considered the following alternatives:

- A. Model the 360/50, and validate the model
- B. Model the 360/30, which couldn't be validated in detail
- C. Do both A and B

The first alternative was rejected since it would not permit the comparison desired by the Army between the IBM and Honeywell computers. The advantage to alternative A was that we would produce a model which could be validated in detail, thus demonstrating the effectiveness of our methodology. We did not have the time to produce three models so alternative C was also rejected. Instead, we concentrated on getting as much data as possible from a SIDPERS cycle running on a 360/30.

Section 4 details our data collection activities for the SIDPERS Basic Cycle. The next section presents our approach to modeling hardware/software systems for performance evaluation, ranking and selection.

3. THE IPSS APPLICATION PROCESSING SYSTEM METHODOLOGY (IAPS)

3.1 THE IAPS MODELING PERSPECTIVE

The problem addressed in this reserach is the design and implementation of a model building methodology to assist in the evaluation of computer hardware/software systems. The goal is a methodology with the widest possible applicability to the user community. Therefore, the IPSS design goals have been adopted, namely:

1. Breadth of Applicability -- the ability to model the behavior of contemporary and foreseeable system architectures and operating environments;
2. Functional View of Systems -- the ability to identify and characterize system components and activities based on their function, independent of a particular architecture or environment;
3. Top Down, Modular Model Synthesis -- the ability to model to a level of detail commensurate with research objectives;
4. Expandable Structure -- the capability to incorporate new, higher level descriptive facilities and performance measures into the methodology and simulation system; and
5. Flexibility of Use -- the ability to be used by a wide spectrum of modelers from the experienced system analyst/designer and researcher to the practitioner and student.

Because of the wide range of knowledge required for modeling computer systems, the IAPS methodology reported in this research distinguishes four distinct modeling functions and provides facilities and tools for each. These functions partition the modeling and evaluation of computer hardware/software systems into a set of activities to be performed by:

1. the User,
2. the Modeler,
3. the Simulator, and
4. the IPSS Analyst.

These activities are summarized in Figure 3-1. As shown, the Modeler is responsible for the creation and maintenance of model libraries, the User for the selection of library members *to synthesize a model*, the Simulator and IPSS Analyst for maintaining IPSS source code and execution facilities. We now define these job functions in more detail.

User - that person or persons whose responsibility is the evaluation of computer hardware/software systems. The user conducts modeling experiments by selecting pre-defined model components from a model library, and selects execution options. The user validates the model, analyzes the simulation results and performs the required evaluation.

Modeler - that person or persons whose primary concern is with the application system to be modeled and with the hardware environment on which it will execute. The modeler builds and maintains

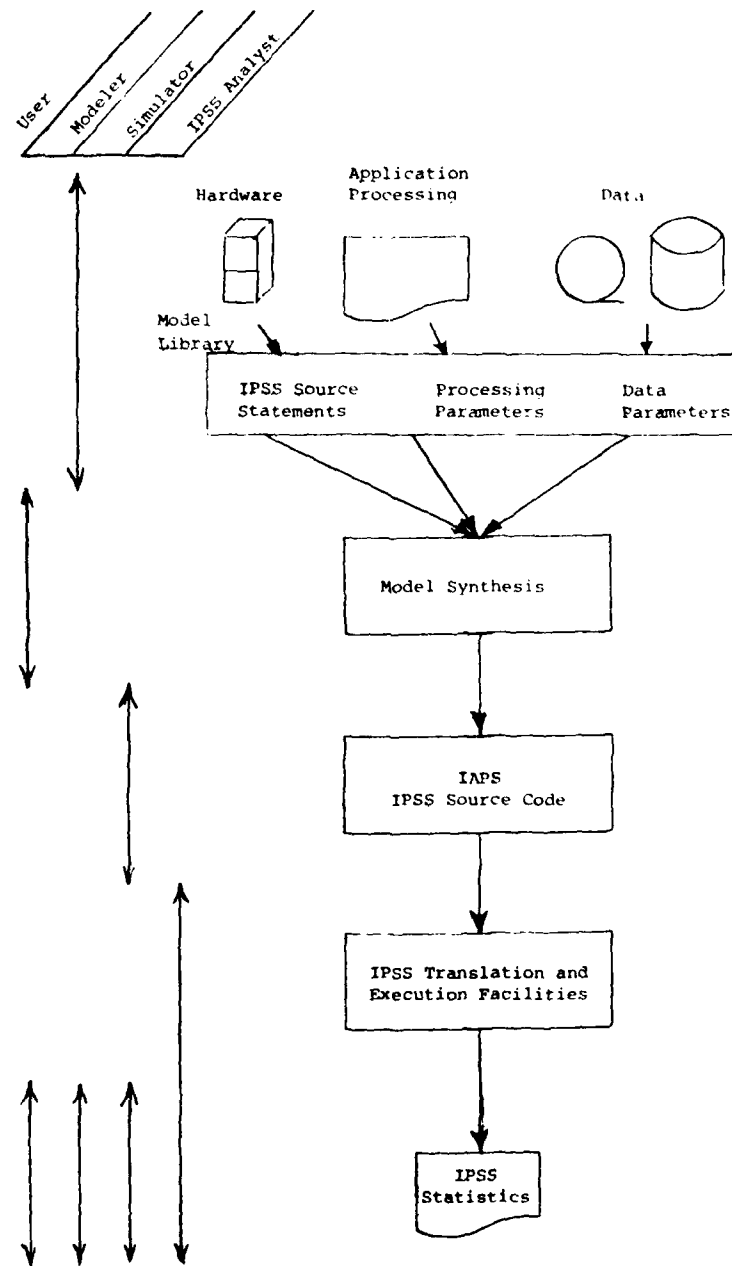


Figure 3-1. Overview of the IAPS Methodology

the model library of software and hardware components. The modeler is not concerned with the structure or execution of the IPSS model, but is concerned with model verification.

Simulator - that person or persons whose primary concern is with the structure and execution of the IAPS model. The Simulator codes user-required special-purpose IPSS routines, incorporates these routines into the model, and verifies their correctness.

IPSS Analyst - that person or persons who have a detailed knowledge of the inner workings of the IPSS simulator. This includes the source language translation process, the simulation driver, facility definitions, and tables.

User level activities were established so that model synthesis and experimentation could be easily accomplished. Hardware characterization and workloads can be changed by the User without any change to the IPSS model itself. This approach assumes a library of computer system characterizations. Our research is the first step in providing an IPSS system library for the User.

The role of the User is distinct from that of the Modeler, Simulator and IPSS Analyst; the major distinction is that the User produces no IPSS source code or workload characterizations. The Modeler and Simulator may be the same person or persons. They must coordinate their activities so that the resulting simulation can be validated. For example, the Modeler describes computer hardware using IPSS statements, but it is the Simulator who describes the sequences of the IPSS simulator's acquisition, use and release of this hardware.

The Modeler, Simulator and IPSS Analyst each share a common set of modeling activities. As summarized in Table 3-1, these activities are: Hardware characterization, software description, sequence of activities, data description and model verification. As shown in the Table, the Modeler's role is independent of any procedure oriented code. The Simulator has the responsibility for maintaining the IAPS source code, and relies on the IPSS Analyst for special functions or requirements.

The remainder of this section is organized as follows. Section 3.2 outlines the User activities, Section 3.3 presents the Modeler view, Section 3.4 discusses the Simulator view and relates it to the Modeler. The IPSS Analyst function is presented in Section 3.5.

3.2 THE IAPS MODELING APPROACH: THE USER'S ROLE

The user is defined to be that person or persons with overall computer system evaluation responsibility for a given project. As shown in Figure 3-2, the user accepts and clarifies a set of evaluation requirements and produces evaluation documentation through:

- o interaction with the Modeler function,
- o interactive model synthesis,
- o validation of model results, and
- o analysis and evaluation of computer hardware/software systems.

The User interacts with the Modeler in order to ensure that the desired model library members are present for the model synthesis phase. The Modeler may be required to change the existing library members, add new ones, or to add capabilities to the IPSS model itself (such as DBMS processing) in order to satisfy the User's modeling requirements.

Table 3-1. Modeling Activities

Modeling Activity	Modeling Perspective		
	Modeler (Section 3.3)	Simulator (Section 3.4)	IPSS Analyst (Section 3.5)
1. Hardware Characterization	IPSS Hardware-oriented facilities	Channel program service	Facility table definitions
2. Software description	Execution group statement	IPSS endo services	IPSS statement parsers
3. Sequence of activities	Ordering of execution group statement	IPSS services, Equates	IPSS driver, Current and future event queues
4. Data description	User and system file tables	IPSS data base component	Logical record to physical address translation
5. Verification	Queuing and Utilization of hardware facilities	Sequence of events in the model	Random number generator, etc.

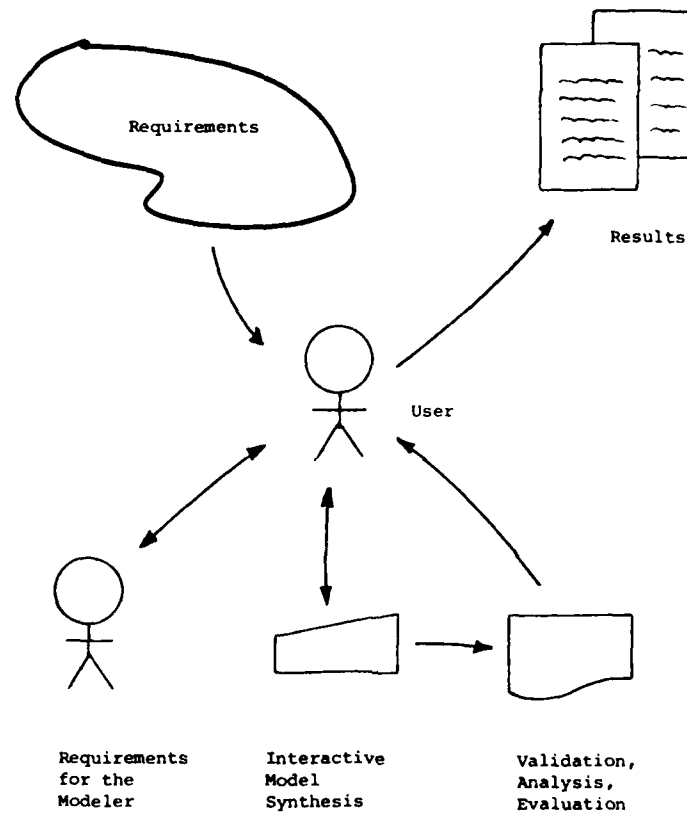


Figure 3-2. User's Role in the IAPS Methodology

The next step in the User process is interactive model synthesis. This produces an execution-ready model through the selection of a priori defined model components from the model library. This process is illustrated in Figure 3-3. We have implemented interactive model synthesis, and used it to produce the results reported in this research. An example of the User-computer interaction sequence is presented in Figure 3-4.

We designed, but did not implement (due to lack of time) a more elaborate model synthesis procedure which would allow the user to modify some of the existing library members in order to tailor them for specific processing needs. As shown in Figure 3-5, we envision that the software processing and data base description members of the model library could be so tailored. This would require more user interaction than now required but would enhance flexibility. This approach is further discussed in Section 9.

3.3 THE IAPS MODELER VIEW OF COMPUTER HARDWARE/SOFTWARE SYSTEMS

Figure 3-6 shows that the Modeler's responsibilities include the preparation of data for input to the model, and the verification of resultant simulation statistics. Input data preparation involves the following:

1. Description of the system hardware,
2. Description of the applicaiton processing workload
3. Description of the characteristics of the data files used by the application, and
4. Representing these descriptions according to the IAPS methodology specifications and storing them in the IAPS model library.

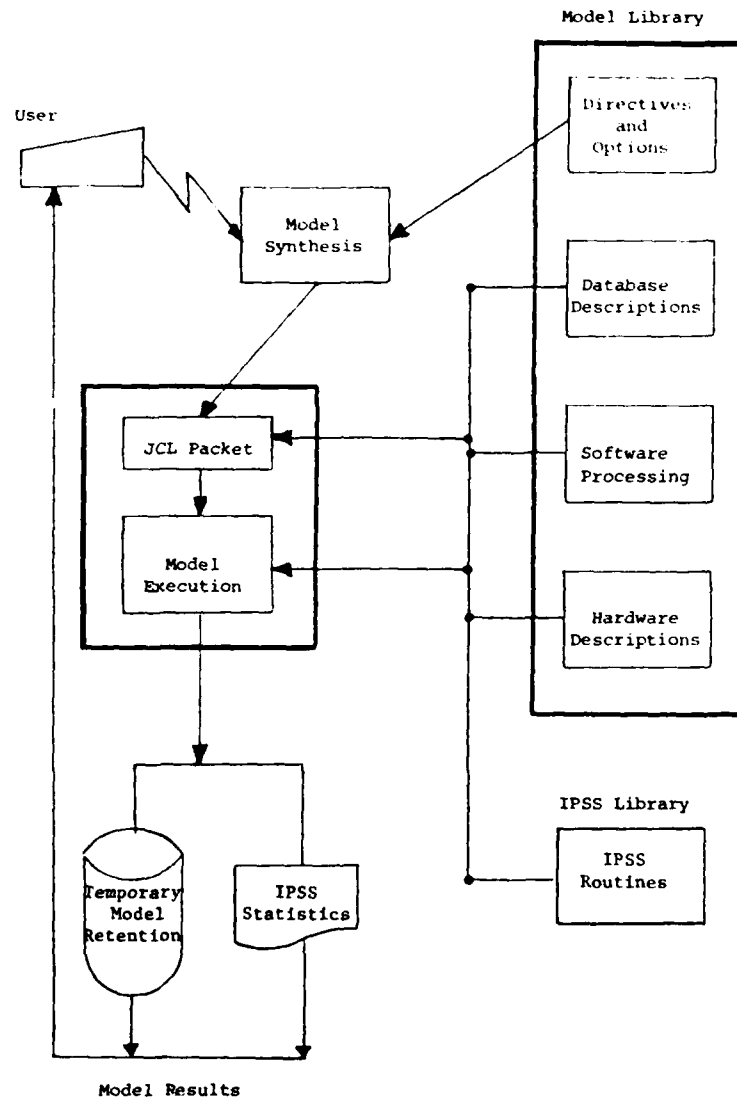


Figure 3-3. User's View of IAPS Model Synthesis and Execution

NEOIRP01

WHAT IS THE MODEL# 30 OR 477:30

WHAT IS THE NAME OF THE IRP01 MODEL-COMPONENT? CTDM30

WHAT IS THE NAME OF THE SYSTEM FILE TABLE? SVIFTAB

WHAT IS THE NAME OF THE USER FILE TABLE? UFLTAB

WHAT IS THE I/O PROCESSING TABLE? TESTPROC

WHAT IS THE 10 DIGIT RANDOM NUMBER SEED? 0123456789

DO YOU WISH TO SAVE THE LOAD MODULE (Y OR N)? N

DO YOU WANT A FORTRAN SOURCE LISTING (Y OR N)? N

JOB 4279 NEOMOD30 ON INTRIP BROWNH WITH
READY

Note: Underlined entries are User input or response.

Figure 3-4. Example of Interactive Model Synthesis

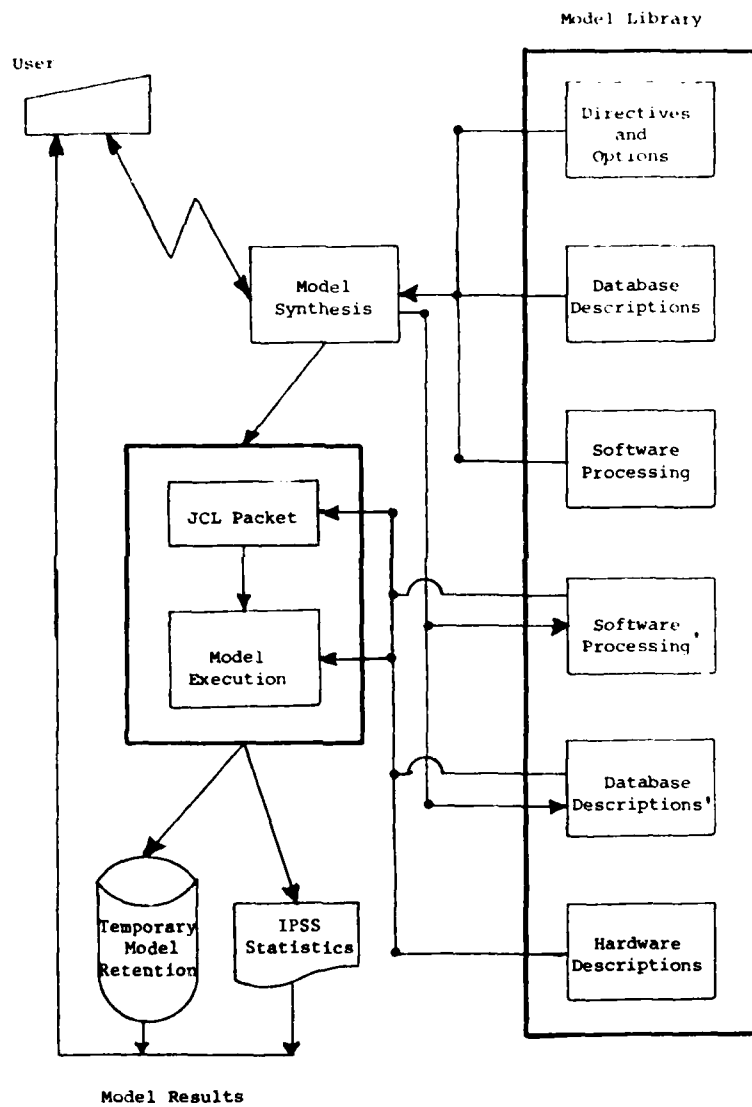


Figure 3-5. User's View of IAPS - Proposed

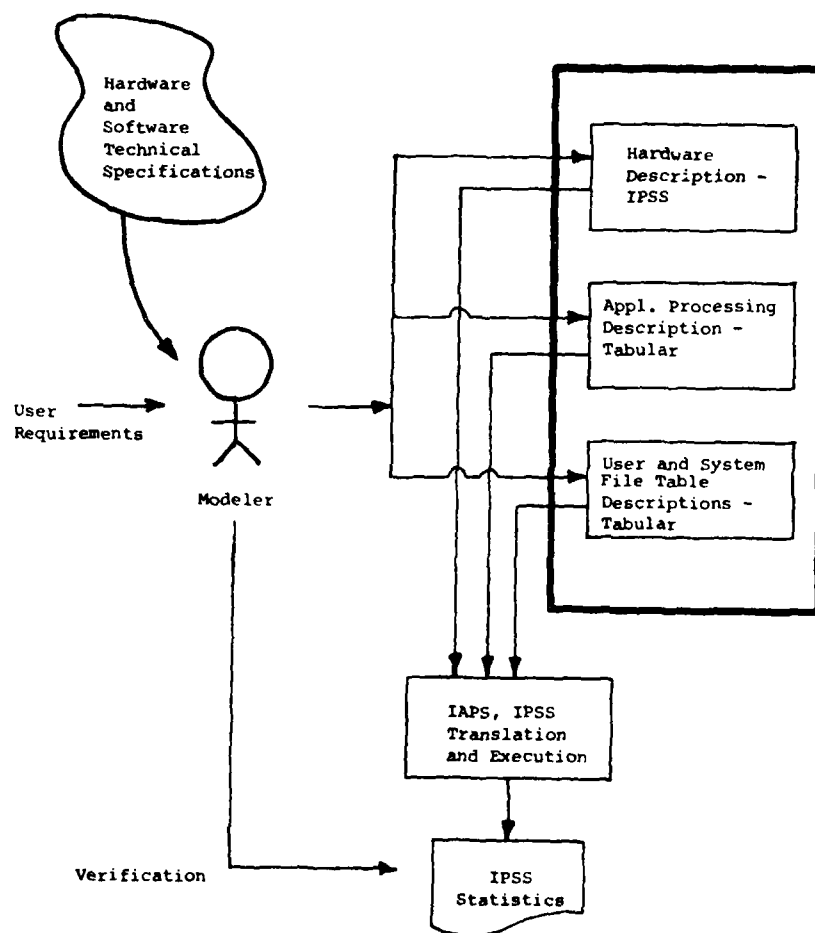


Figure 3-6. Modeler's Role in the IAPS Methodology

An overview of these Modeler activities is now presented. Details are found in the Appendices as noted in the text.

Description of System Hardware

The Modeler is responsible for identifying the basic types of computer devices for the system being modeled. As shown in Table 3-2, the devices primarily reflect the computer's mainframe and secondary storage subsystem. For each device identified, the modeler provides a detailed functional specification which indicates capacity, speed, and special features. A list of the type of data collected for disk, drum, tape, and unit record devices is given in Table 3-3. This type of data is usually readily available in vendor's technical system reference manuals. The Modeler then encodes this data into IPSS statements in a straight-forward way. Examples of these IPSS statements are found in Appendix B.

Description of Application Processing Characteristics

The application workload and its data files are characterized by two types of tables which are prepared by the modeler. These tables are called the Application File Table (AF Table), and the Application Processing Table (AP Table). The Application File Table gives detailed information about the files being processed from the application program point of view. The Application Processing Table gives, in outline fashion, a step by step description of application processing.

The Application File Table (AF Table)

The AF Table describes the characteristics of files as known by the application program. Each entry in this table describes a single file and contains: a file-identifier, the logical record length and

Table 3-2. Modeler Checklist for Computer System Hardware

Device Type
CPU
Main memory, cache
Channels (multiplexor, selector)
Disk Units, Disk Controller
Tape Units, Tape Controller
Drum Units, Drum Controller
Operator's Console
Line Printer
Card Reader
Card Punch

Table 3-3. IPSS Data Required to Model I/O Devices

Device Type	Data
<u>Disk, Drum</u>	number of packs per control unit number of cylinders per pack number of tracks per cylinder maximum track capacity maximum block size allowable for the device rotational speed data transfer rate cylinder access times
<u>Tape</u>	number of tape units per control unit tape recording density tape speed (reading/writing) inter-block gap size maximum block size recorded on the tape tape start-stop time forward erase length rewind rate
Unit Record (Card reader, punch, operator's console, etc.)	maximum block size transmission mode transmission rate

block size, and the number of records processed. An example is given in Figure 3-7. This example shows two files, one an unblocked card-image file of 554 records which is identified through the comment as SIDPERS file COOAAC. The other file, CICAAC, contains 987 506-byte records. Note that the block size specified in the AF Table is the unit of I/O for the application program and need not represent the secondary storage block size.

The Application Processing Table (AP Table)

The AP Table mimics the I/O processing done by an application program. Each table entry consists of two records, first a processing specification record, followed immediately by a processing definition record.

The specification record identifies the type of processing, (D for any delay due to the operator, I for input, P for CPU activity, and O for output). The "D" definition record quantifies the delay; the "I" and "O" definition records specify: the file, a concurrency index, random or sequential processing, and percentage of file processed; and the "P" definition record specifies the type of activity engaged in by the application program, such as EDIT, SORT, or REPORT.

Figure 3-8 depicts a typical example of a job step (for example, SIDPERS). First, there is a delay of 10 to 15 seconds due to operator responses to console messages, or tape mounts. Then all of file 01, and 50% of file 10 are read concurrently, file 01 sequentially and file 10 randomly; one of the files is edited, and the output is sent to file 04. Finally, 5% of file 10 is rewritten after all other processing is complete. Note that the order of application record processing is

File id	L.Recl	Block Size	# Records	Comments
01	80	80	554	Card Image Input - C00AAC
10	506	506	987	Edit Table File - C1CAAC

Figure 3-7. An Example of the Application File Table

```

*
* A Typical Jobstep in SIDPERS
*
D      1000.    15000.    1.0    TAPE MOUNT
*
I      01 X S 100.    COOAAC
*
I      10 X R 50.    C1CAAC
*
P      X      EDIT    CPU PROCESSING
*
O      .04 X S 100.    B1AAAC
*
O      10 Y R 10.    C1CAAC

```

Figure 3-8. An Example of the Application Processing Table

determined by the concurrency indicator (the "X" and "Y" in Figure 3-8). The "X" indicates that files 01, 10 and 04 are processed concurrently, (i.e., read one record of file 01, one of file 10, write one to file 04, then repeat until all three files are exhausted). The "Y" of Figure 3-8 indicates that file 10 is written after the complete processing of files 01, 10, and 04. (Any alphanumeric characters, except blank, may be used as a concurrency indicator). As also shown in this figure, comments may appear on the right hand side of any data card, and on any "comment" card (which is designated by an asterick in the first column).

Description of Database Characteristics

The term database is used here to mean the data files managed by the hardware system's data files. Those files required by an application must be characterized by the Modeler and the results placed in the System File Table.

In this table, each record gives secondary storage information about a single file. Each file is assigned a volume type (disk, tape, or console) and a volume number. The logical record length, blocksize, and file size (number of logical records) are recorded. Disk file information includes the extent type (index (I), primary (P), or overflow (O)) the percentage of records on the primary extent (%PE), the number of secondary extents (#SE). If the file placement is known, it is given in terms of low and high cylinder and track addresses. (If unknown for disk files (U), the file is placed randomly on the volume during IAPS simulation). Finally a comment field is provided as an aid to the modeler. Provision is made to define VTOC files (V),

sort work files, and messages to and from an operator's console.

For detailed formatting information, see Appendix F.

The examples in 3-9 are typical. System file 01 has 554 unblocked records with an LRECL of 80. It resides on tape unit T01 with an LRECL and blocksize of 80, and an "unknown" placement (U), which for tape files means that the file begins at the beginning of the reel. Note that the modeler has used comments to identify the file as COOAAC -card image input. User file 10 (the third line of Table 3-9), is also unblocked with an LRECL of 506; it has 987 records in its prime extent (P), which resides on disk unit D03, with allocated space from cylinder 153 track 0, to cylinder 170, track 19. This file is an ISAM file and thus has an index extent which resides on disk unit D02, giving among other things, its known placement.

3.4 THE IAPS SIMULATOR VIEW

The IPSS Simulator function requires a person or persons who are knowledgeable programmers and analysts of the IPSS language and execution facilities. The basic simulator role is to augment the IPSS model structure we have provided in order to be responsive to changing Modeler requirements.

A Simulator overview of the IAPS methodology is given in Figure 3-10. This figure shows the interaction among three components of IPSS: The Exogenous Event Stream Component. The IPSS Define Model Component is represented by the Equates between the two latter components.

We have completed the simulator function for the present IAPS methodology. A large class of computer systems can be simulated and

File Identification	Storage Media	Logical Record Length	Block Size	Number of Records	% of Records in Primary Extent	Number of Secondary Extent	Known or Unknown Placement	Extent Type	Comments
01	T01	80	80	554			U		
10	D02	22	22	18	100	0	K	I	199-13 199-13 CICAAC-INDEX
10	D03	506	506	987	100	0	K	P	153-00 170-19 CICAAC-PRIME
23	D02	256	256	17			K	V	000-00 000-19 VT0C-D02

Figure 3-9. An Example of the System File Table

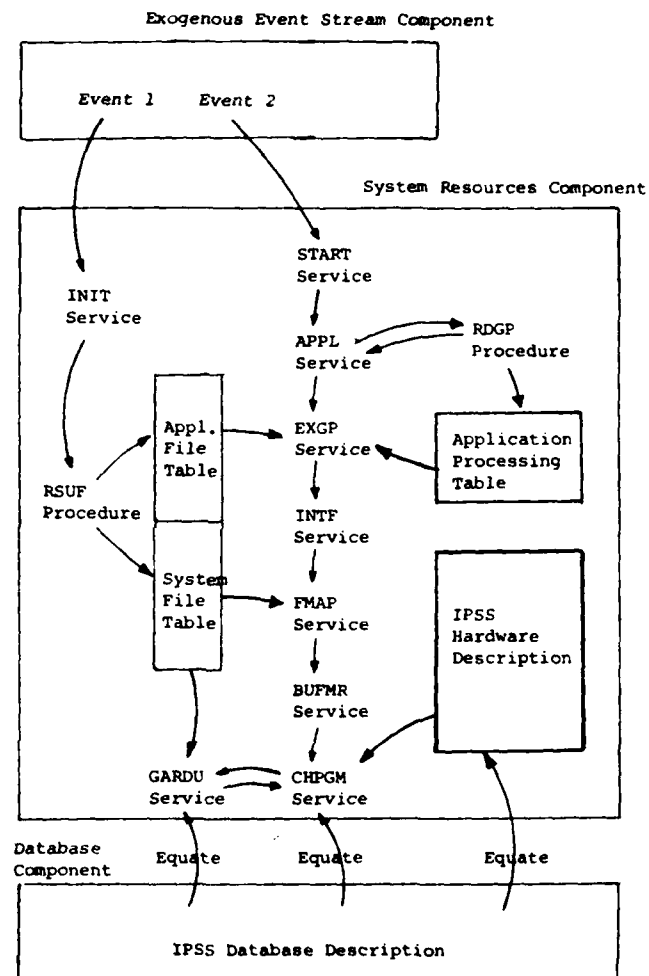


Figure 3-10. Overview of the Service Hierarchy in the IPSS Model -
The Simulator's View of the IAPS Methodology

evaluated without any further Simulator activity. The Simulator is required if a system outside the scope of the present IAPS is to be modeled. Examples of such systems are: Database management systems, teleprocessing, distributed systems and operating system task management.

3.5 THE IPSS ANALYST VIEW OF COMPUTER SYSTEMS

The IPSS Analyst is a specialist in the IPSS Modeling and Execution facilities (see Appendix A). The IPSS Analyst view of the simulation process is represented in Figure 3-11. This role requires a knowledge of the details of the IPSS translation process, the simulation nucleus, and facility definition tables. The need for the IPSS Analyst will be further reduced over time as IPSS evolves into a more fully developed product. We required the type of knowledge represented by the IPSS Analyst role only a few times during the course of our project. Examples of what we required (and easily ascertained through source listings) were IPSS statement options, random number generation algorithms, and facility table value offsets.

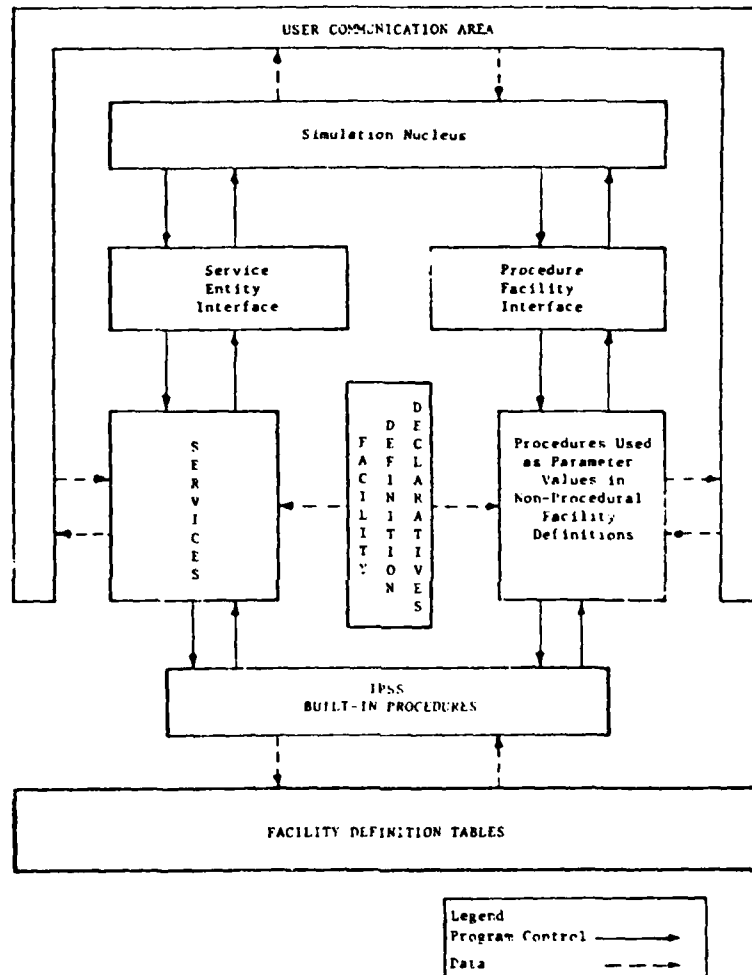


Figure 3-11. IPSS Program Control and Data Interfaces - The IPSS Analyst View of IPSS (DEL78a)

4. MODELING SIDPERS USING IAPS

4.1 SELECTION OF A SIDPERS SUBSET

SIDPERS is a standard, automated integrated personnel system designed to provide personnel information systems support at division, installation, brigade, battalion and unit levels (SID76). SIDPERS performs four major functions in support of Active Army personnel and organizations:

1. Strength accounting,
2. Organization and personnel recordkeeping,
3. Information exchange with other automated systems, and
4. Command and staff reporting.

A SIDPERS activity is designed to support a data base of computer files for up to 50,000 personnel and 1,000 organizations.

SIDPERS software consists of five DOS-E jobs:

- o AACRO1 - Labels and Assignments
- o AACRO2 - SIDPERS Basic Cycle
- o AACRO3 - SIRCUS
- o AACRO4 - SIDPERS Back-up Cycle
- o AACRO5 - SIDPERS Recovery Cycle

The focus of our project was on the SIDPERS Basic Cycle, Job AACRO2. This job consists of 19 job steps which proceed from editing functions, through file update, to reporting. Since the project duration and objectives did not permit the modeling and evaluation of all of SIDPERS, a subset of programs was selected with the assistance of USACSC Quality Assurance personnel (WHI79).

The editing programs, two of the master file update programs and one report preparation program were selected from all the programs in the SIDPERS basic cycle. Each editing program was a single job step and thus some validation data could be obtained. The selected update and report preparation programs were, however, single phases loaded and executed dynamically as part of a larger job step. While the modeling of the logic of these programs was not a problem, obtaining validation data for these phases was not possible. In addition, the modeling of the entire job step in which these phases were located would be almost as difficult, again, for lack of adequate validation data. Thus, the selected update and report preparation programs were not modeled.

The programs we modeled represent transaction classification, sorting, and validity editing; and incorporate concurrent direct and sequential access to disk files and sequential access to tape files. These programs are:

PIAAACA - transaction classification and scheduling,

PIBAACS - transaction sort,

PICAAC - transaction validity editing, and

PIGAAACS - sort and update "queue" production.

For convenience and readability, these programs will be referred to as PIA, PIB, PIC, and PIG, respectively.

4.2 MODELING THE SIDPERS SUBSET

Once this subset of the SIDPERS programs was identified, we obtained current COBOL source listings, the DOS version of the SIDPERS

Operations and Scheduling Manual (SID79), and the SIDPERS Basic Cycle JCL listing. We analyzed these sources in order to obtain basic file data which is constant to any SIDPERS processing cycle. The type of data we obtained were file names, type of file (e.g., ISAM, sequential), use of file (input, output, both input and output), record processing mode (sequential or random). The details of our findings are summarized in Appendix C.

Next, we determined that the data we required for validating a model of SIDPERS was available from four sources, namely:

1. GRASP Accounting,
2. SYSLIST,
3. Operator Console Log, and
4. DITTO Utility.

The type of data provided by each of these sources is summarized in Table 4-1.

We visited the Ft. Stewart Division Data Center on August 2nd and 3rd, 1979, and observed the computer operation during SIDPERS Basic Cycle processing. We obtained computer listings for the data sources listed above. Table 4-2 presents a summary of the data we collected at Ft. Stewart for the first four job steps of the SIDPERS Basic Cycle on August 2, 1979, and indicates the source of the data items.

The following is a discussion of these data sources in more detail.

GRASP

GRASP provides a wealth of accounting data which is extremely useful in validating simulation models. For completeness, a list of the type of data available through GRASP is provided in Appendix G.

Table 4-1. Sources of System Data

1. GRASP Step Accounting

CPU time
Wait on operator time
Job duration time
Interference duration time
I/O wait time
I/O device usage time
Start I/O counts
Time waiting for and using the LTA
SYSRES usage time
Channel activity time

2. SYSLIST

Gives job step start and stop time
Number of records sorted
Some file counts

3. Operator Console Log

Number and length of console messages

4. DITTO Utility

Record counts
Type of records processed

Table 4-2. 2 August 1979 Ft. Stewart Record Processing for Programs
PIA Through PIG

File Name	Media	Input/ Output	Concurrently Processed with	% File Processed	Number of Records	Source of Record Count Data
<u>PIA</u>						
COOAAC	Tape	I		100	554	Card count
BIAAAC	Tape	I		100	2111	BIAAAC out + 31 Grade Changes
COOAAC	Tape	I		100	661	Tape DITTO
AIAAAC	Tape	O		100	1249	SYSLIST sort count in P1B
EIAAAC	Tape	O		100	1171	Tape DITTO
BIAAAC	Tape	O		100	2080	Tape DITTO
CICAAC	Disk	I/O			987	Program source and # transactions
XUTAAC (X=A,B,C, E,F,G,H)	Disk	I		0	-	Program source listing
<u>P1B</u>						
AIAAAC	Tape	I		100	1249	SYSLIST
BIAAAC	Tape	I		0	-	Monthly only
CICAAC	Disk	I/O		.3	987	
AIBAAC	Tape	O		100	1249	SYSLIST
BIAAAC	Tape	O		0	-	(see above)
SORTWK1-5	Disk	I/O				Computed

Table 4-2 Continued.

File Name	Media	Input/ Output	Concurrently Processed with	% File Processed	Number of Records	Source of Record Count Data
<u>PLC</u>						
ALBAAC	Tape	I		100	1249	SYSLIST
CLCAAC	Disk	I/O		125	987	Source program and input transactions
ALCAAC	Tape	O		100	1249	SYSLIST
<u>PLC</u>						
ALCAAC	Tape	I		100	1249	SYSLIST
RLCAAC	Tape	I		100	50	
CLCAAC	Disk	I/O			987	
SORTWK1-6	Disk	I/O				Computed
X1GAAC (X=A,B,C, E,F,G,I,J, K,M,N,Q,R)	Disk	O		(A) 0	-	SYSLIST
				(B) 0	-	SYSLIST
				(C) 100	30	SYSLIST
				(E) 100	1210	SYSLIST
				(F) 100	7	SYSLIST
				(G) 0	-	SYSLIST
				(I) 0	-	SYSLIST
				(J) 0	-	SYSLIST
				(K) 100	2	SYSLIST
				(M) 0	-	SYSLIST
				(N) 0	-	SYSLIST
				(Q) 0	-	SYSLIST
				(R) 0	-	SYSLIST

GRASP, however, was of limited use to us since GRASP step accounting was not available at the Ft. Steward Division Data Center. Table 4-3 summarizes the data we obtained from GRASP for the August 2, 1979, execution of the SIDPERS Basic Cycle. As shown in this table, GRASP job accounting statistics did not provide us with any useful data for programs P1A through P1G. Since the cycle we observed was initially cancelled in program P1G, we were able to use the GRASP CPU time of approximately twelve minutes as an estimate of the complete P1A through P1G CPU time.

SYSLIST

SYSLIST was of value in determining file record counts only when the job contained a sort. Program P1B and P1G sort entire files and the number of records sorted is reported on the SYSLIST.

Operator Console

The operator console log did not provide us with any data on the number of records processed. However, we observed that, because of the amount of time spent displaying and responding to messages, the operator's console was a more important element in the system from a performance perspective than we originally anticipated.

Tape DITTO

By far the most useful method of determining the number of records processed on a per file basis is the Tape Utility DITTO. This utility simply lists the entire file, allowing not only an accurate count but also insights into the types of data being processed. We obtained DITTO listings of the transaction input files and the stacker files.

Table 4-3. SIDPERS Basic Cycle, 2 August 1979, Ft. Stewart (Extracted from GRASP Accounting Reports)

Activity	Complete Cycle	P1A through P1G cancel	P1A through P1G complete
Elapsed time	4/33/27	38/54	17/50**
Non-MPS time	4/25/23*	38/53	N/A
CPU time	2/36/12	11/56	N/A
Pages spooled	165	11	N/A
Pages loaded	4443	363	N/A
Transient Area time			
Wait	10	0	N/A
Use	1/15/34	26/05	N/A
RES I/O	31622	1893	N/A
Operator console time	42/28	23/10	2/27***
* Does not include 07/11 restart time ** from SYSLIST *** from Console log			

5. MODELING TWO COMPUTER HARDWARE ARCHITECTURES IN IAPS

5.1 IPSS HARDWARE CHARACTERIZATION

This section discusses the modeling of the IBM 360 Model 30 computer (CS₃) and the Honeywell Series 60 Level 6 Model 47 minicomputer (DAS₃) systems. A computer architecture is typically represented in IPSS by characterizing the following:

1. the hardware devices, their capacities and processing characteristics,
2. the interconnections among the hardware devices, and
3. the operating system.

Hardware Devices, Capacities, and Processing Characteristics

The block diagram for the two modeled systems are shown in Figure 5-1 and 5-2. The connecting edges between primary and secondary storage represent the paths along which data is transferred. Note that in the 360/30, the dual channel tape controller allows either channel 1 or channel 2 to complete an I/O request. Thus, there are two paths to the tape units and one to the disk. We assume that channel 1 will be used to access a tape unit when channel 2 is busy.

The focal point of the IPSS modeling of these systems is on the secondary storage subsystem. We examined technical specifications provided by the respective vendors and extracted performance characteristics and capacities for both the direct access storage devices and the magnetic tape units. These characteristics are reported in Tables 5-1 and 5-2 respectively. This data was incorporated into

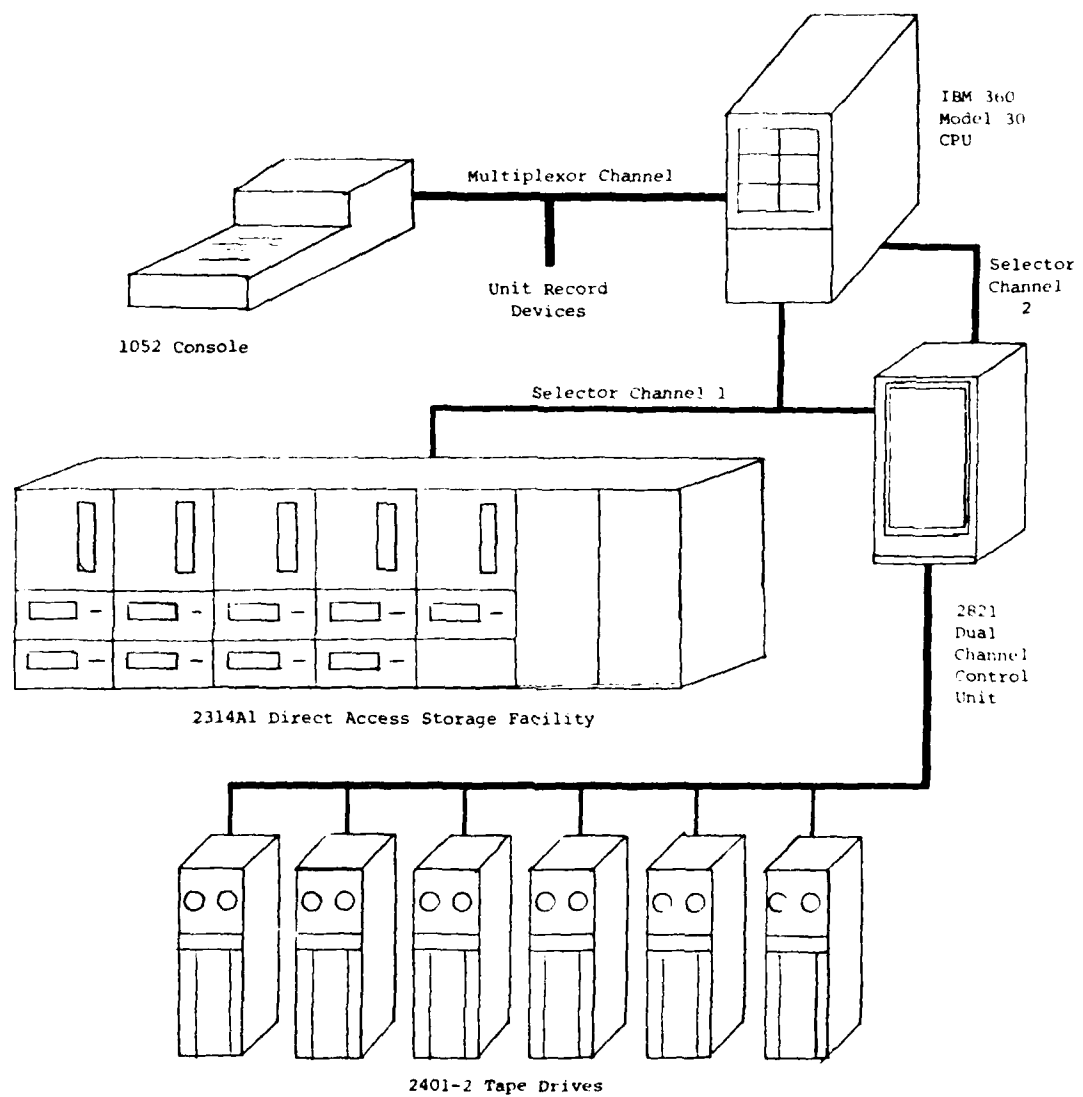


Figure 5-1. Typical CS₃ Hardware Configuration

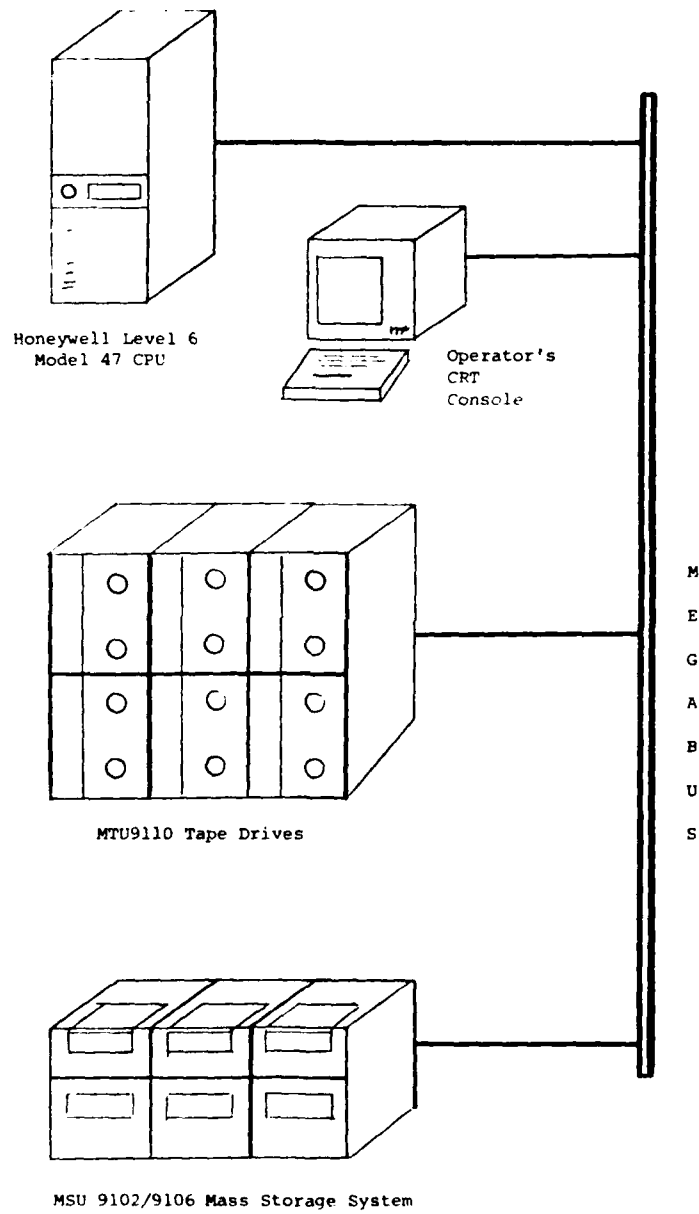


Figure 5-2. Honeywell Level 6 Architecture - DAS₃

Table 5-1. Direct Access Storage Devices

Physical Characteristics	Disk Units		
	IBM 2314A	IBM 3330	Honeywell MSU 9102/9106
Drives per unit	8	8	3
Bytes per unit	233.4M	800M	201M
<u>Speed</u>			
Average Seek	60 ms	30 ms	30 ms
Average Rotational Delay	12.5 ms	8.4 ms	8.33 ms
Data rate (kilobytes per second)	312	806	1,200
<u>Capacity</u>			
Cylinders per pack	200	404	823
Tracks per cylinder	20	19	5
Tracks per pack	4,000	7,676	4,115
Bytes per track	7,294	13,030	16,384*
Bytes per cylinder	145,880	247,570	81,920
Bytes per pack	29.18M	100M	67M
*Based on 64.256 byte-sectors per track			

Table 5-2. Magnetic Tape Units*

Physical Characteristics	Tape Units		
	IBM 2400-1 Model 5	Honeywell MTU 9109	Honeywell MTU 9110
Drives	6	2	6
Tracks	9	9	9
Density	800/ <u>1600</u>	800/ <u>1600</u>	800/ <u>1600</u>
Inter-block gap (inches)	.6	.6	.6
Block length	-	2048	-
<u>Speed</u>			
Read/write (inches per second)	75	45	75
Rewind rate (inches per second)	350	200	250
Data transfer rate (bytes per second)	120k	36k/ <u>72k</u>	60k/ <u>120k</u>
Start/stop time	13 ms.	8.33 ms	5 ms
*Where more than one characteristic is listed, the underlined number was included in the model(s).			

IPSS models through IPSS hardware-facility statements. A sample of these statements is given in Appendix B.

Interconnections Among Hardware Devices in IPSS

In the IPSS System Resources component, device characteristics are associated with an access mechanism and volume. However, channels, control units, and the CPU are independent, unrelated facilities. These facilities are interrelated in IPSS models by a service which represents a channel program. This service, usually called CHPGM, is almost a standard part of every IPSS model and plays a central part in the IAPS methodology. Its function is to generate a physical (device) address and to seize, use and release all the facilities (e.g., CPU, channel controller, access mechanism, volume) in the path from the CPU to the secondary storage device in order to simulate a data transfer. The IPSS CHPGM service is listed in Appendix B.

Operating System Representation in IPSS

In IPSS, an operating system is represented by one or more services which simulate job scheduling, task management and resource allocation activities. We investigated but did not represent the operating system functions in either the IBM or the Honeywell model. The reason is that we did not have time to analyze these function, or model them, in sufficient detail to warrant their inclusion in the models.

However, our investigation revealed that the IBM 360 Model 30 supports a Disk Operating System (DOS) with the following major support packages:

- o GRASP accounting package,
- o DYNAM/T tape manager,
- o ADAS disk manager, and
- o SYNC SORT sort package.

We attempted to ascertain how these packages interact with DOS and under what conditions they request I/O. The next step would have been to represent processing, resource allocation, I/O characteristics, and dispatching of each of these packages (including DOS) in one or more IPSS Endogenous Services. Following this, we would include these services at the appropriate place in the IPSS model (i.e., at the INTF service), then verify and validate the resulting model.

5.2 ARCHITECTURAL VARIATIONS

For convenience in referencing the hardware systems that we modeled, we designated the model of the IBM 360 Model 30 as Model A1, and will refer to the model of the Honeywell Level 6 minicomputer as model B1. In addition, we considered three variations of model A1 and one variation of model B1 in order to demonstrate the capabilities of IPSS and the IAPS methodology.

As shown in Table 5-3, the variations on model A1 are the replacement of the 2314 disk unit with a 3330 disk unit (A2), the replacement of the 14 character per second operator console with a 960 character per second console (A3), and both of the above replacements (A4). These experiments were designed based on observations of the current 360 Model 30.

Table 5-3. Hardware Differences

Model Designation	Summary of Architecture Differences
A1	Standard 360 Model 30 - o 14 characters per second operator console - o 2314 direct access storage facility
A2	360 Model 30 - o 14 characters per second operator console - o 3330 direct access storage facility
A3	360 Model 30 - o 960 characters per second operator console - o 2314 direct access storage facility
A4	360 Model 30 - o 960 characters per second operator console - o 3330 direct access storage facility

Table 5-3 Continued.

Model Designation	Summary of Architecture Differences
B1	Standard Honeywell Level 6 Model 47 o 6 MSU9106 disk drives o 6 MTU9110 tape drives
B2	Honeywell Level 6 Model 47 o 3 MSU9106 disk drives o 2 MTU9109 tape drives
*All 360 Model 30 architectures had six 2400-1 Model 5 tape drives. All Honeywell Level 6 Model 47 architectures had a 960 character per second operator console.	

The variation on model B1 was the deletion of three disk drives and the replacement of the 6 MTU 9110 tape units with 2 MTU 9109 tape units.

Table 5-4 shows the performance characteristics of these architectural variations relative to model A1 (the standard 360 Model 30). Model B1 is clearly superior to A1 in every way except tape concurrency (each has six tape drives), and all the variations show at least one area of superiority.

6. OVERVIEW OF IAPS MODEL STRUCTURE AND EXECUTION

An IAPS model consists of a collection of IPSS service facilities whose invocation sequence is hierarchial. Figure 6-1 depicts the relationships among the services and indicates their generic function. As shown in the Figure, the arrival of an application job on a computer is represented by the START service. Several different applications could be started simultaneously and, if so, would compete for systems resources (such as the operating system, main memory, data channels). In the models we synthesized, only one application was started, namely SIDPERS. The START service invokes the application processing service APPL and waits for its completion. The APPL service determines the processing required for an execution group (DOS job step), invokes the EXGP service to perform this processing and waits for its completion. The EXGP service represents the processing performed by a user-determined unit of work. This service is driven by the values provided by the RDGP procedure. Its main functions are to schedule I/O activities to data files, and to represent CPU processing. I/O is represented by an invocation of the INTF service and a wait for response. The INTF service is essentially a null routine which is a system link to future processing activities (such as DBMS or the operating system). Currently, INTF invokes the FMAP service which represents the mapping of the application files to specific system files which are located on secondary storage. A single application I/O request to FMAP will generate one or more requests for system records. FMAP issues a request for a system

record by invoking the BUFMR service and then waits for a response. BUFMR represents the buffer management function. If a system record is in one of the buffers, an immediate response to FMAP is generated, otherwise the CHPCM service is invoked. CHPCM represents channel program processing: it uses the IPSS data base structure and hardware facilities to generate a hardware address, computes the read/write time, and advances the simulator clock accordingly. Table 6-1 relates the IPSS services identified in Figure 6-1 to the corresponding SIDPERS processing activities.

Model Synthesis

One of the advantages of IPSS in general, and IAPS in particular, is the ease of synthesis of experiments. Figure 6-2 outlines our basic approach to producing models with different hardware architectures. We functionally divided the IPSS models and placed the parts into members of a partitioned data set (member names are in parenthesis in the Figure). We then chose members from each of the following categories to form a complete model:

1. Application processing
2. Architecture
3. Define Model
4. Loading

The Application Processing group is the nucleus of our IAPS methodology. It contains all the IPSS services and facilities of the System Resources Component except for hardware facilities and the channel program service. Also included in this group are the IPSS Exogenous Event Stream component and the Data Base Structure Component.

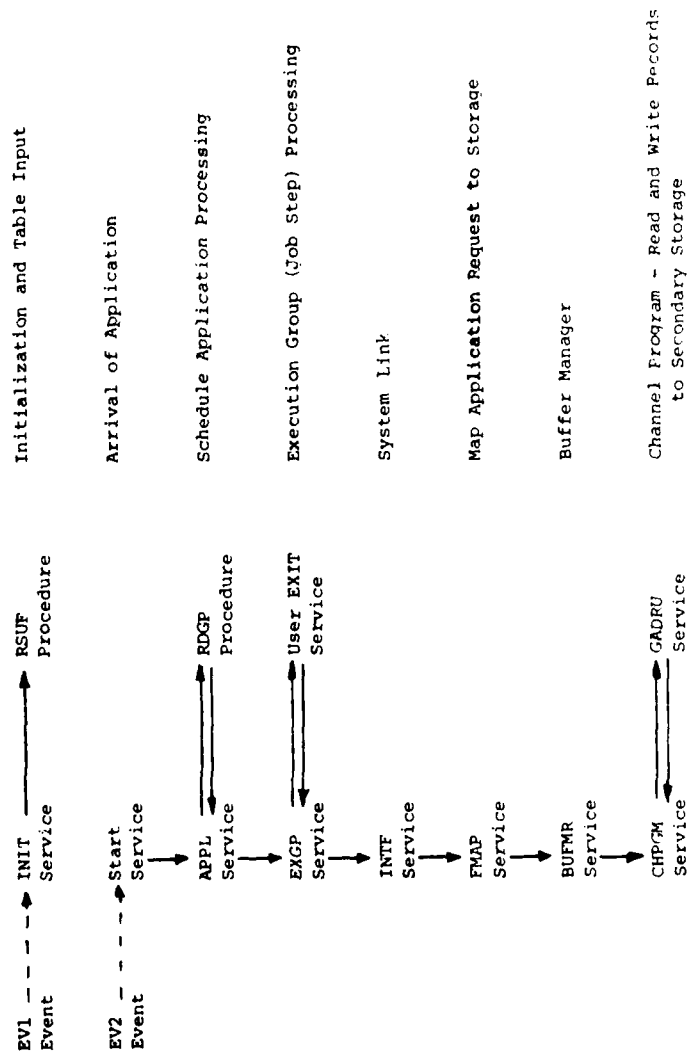


Figure 6-1. Simulator Overview of the IAPs Service Hierarchy

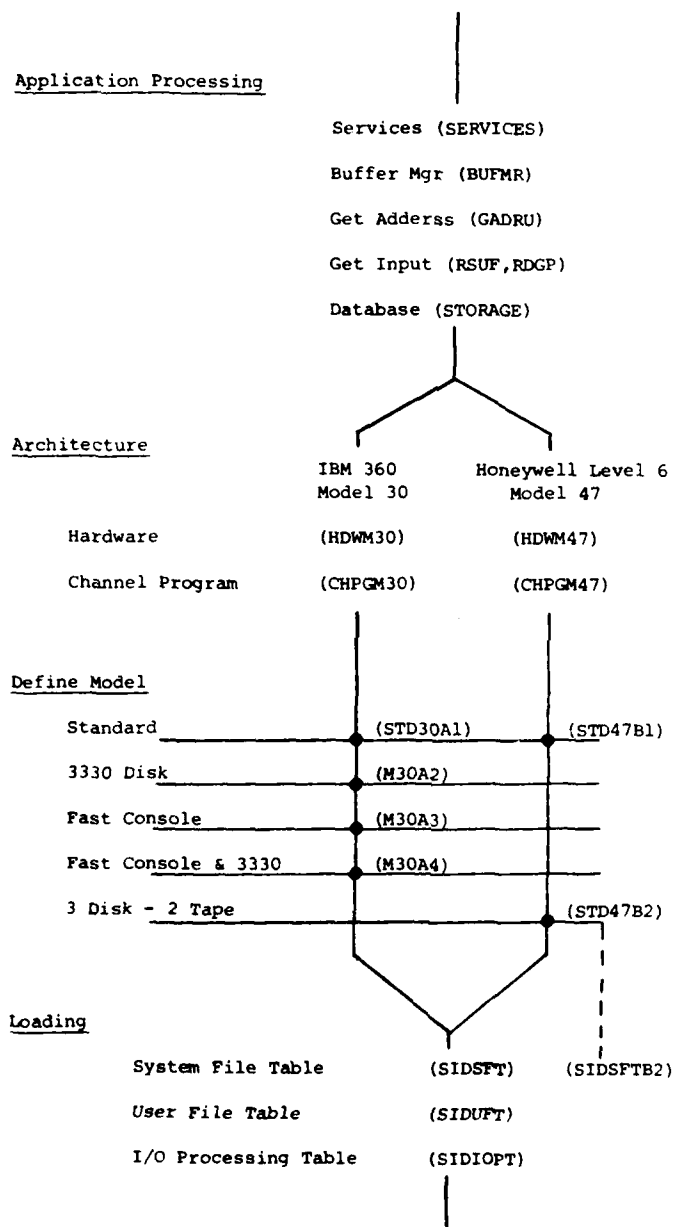


Figure 6-2. Summary of Models Synthesized for Evaluation of Alternate Hardware Configurations

Table 6-1. Index to Modeled SIDPERS Processes

Level	SIDPERS Process	IPSS Endogenous Exogenous Service Name
1	(IPSS initialization)	INIT
2	Arrival of SIDPERS job, job scheduling	START
3	SIDPERS processing	APPL
4	Job Step Execution	EXGP
5	Processing link for OS, DBMS, etc. (future use)	INTF
6	SIDPERS logical record to DOS physical record mapping	FMAP
7	Buffer management	BUFMR
8	Channel program, retrieval of records from secondary storage	CHPGM

The Architecture group contains services and hardware specific to the two generic classes of hardware systems being modeled, namely the IBM 360/30 and the Honeywell Level 6. One set (i.e., hardware specification and channel program) was selected for each execution of the model.

The Define Model group is the IPSS Model component. Each member of this group specifies a hardware alternative. Using these members, we were easily able to replace disk and operator console units in the model and to ascertain the effects on the overall performance of the system. The members of this group were easy to generate and use. Clearly this type of experimentation is one of the primary benefits of modeling in IPSS and using the IAPS methodology.

The last group of members which were selected were from the loading group which represents the external (i.e., SIDPERS) loading on the computer system. These can be easily modified to represent different application processing characteristics.

At the completion of this research project our program library contains members which represent (a) at least eight variations on the basic hardware of the IBM 360/30 (CS₃) system; (b) two variations of the Honeywell Series 60 Level 6 Model 47 (DAS₃) system; (c) a general model of computer software, including submodels of application programs, a buffer manager, and a channel program; (d) tables of data which describe in detail the files used by SIDPERS (Section 4.3); (e) a table which describes the sequence of I/O and processing performed by the first four job steps of SIDPERS, which table is processed by the first submodel mentioned previously in (c); (f) a command procedure in simple question and answer form which allows a user to put together and execute

a complete model from the library members described in a-e, and thus easily to compare design alternatives (Figure 3-4). For a listing of library member names and contents, see Appendix E.

These models were executed on an Amdahl 470 V/6-II with OS/MVS. Each model contained approximately 2800 lines of code (IPSS, Fortran, and comments). Each model required approximately 400KB of main storage and four minutes of CPU time (compilation plus execution). Modeling experiments were facilitated through the creation of load modules which were repeatedly executed. This reduced the simulation run time by approximately one minute.

7. ANALYSIS OF RESULTS OF THE IAPS SIMULATIONS

The ultimate purpose of an IAPS simulation is to present the decision-maker with data which will prove useful in the overall evaluation and comparison of alternative designs. The decision-maker will take many things into account which are not addressed by any simulation, such as the availability of appropriate compilers and the projected cost of maintenance. A valid simulation, however, can provide information which can be obtained from no other source except the much more expensive alternative of running the actual workload on the actual computer system. Examples of such information include answers to the following questions:

1. What is the projected run-time of SIDPERS on the DAS₃ system, and how does it compare to the existing system?
2. Which of the hardware resources is over-utilized, and thus potentially a bottleneck as system workload increases?
3. How will the system respond to an increase in workload over time?
4. How will the system respond if one or more tape, or disk, units become dysfunctional?

For the simulation user to have confidence in the results produced by any simulation, he needs a systematic approach to the validation and analysis of the output statistics of the simulation. It is our purpose in this chapter to outline such an approach in the context of our application of the IAPS methodology to the hardware

comparison of the CS₃ and DAS₃ configurations when run against the same SIDPERS workload. However, we were unable to carry out this approach in its entirety due to lack of data and lack of time.

In the following sections we discuss model verification and model validation, an analysis of the results of simulating the CS₃ and DAS₃ systems in six configurations, and the results of some additional simulations.

7.1 MODEL VERIFICATION AND VALIDATION

Model verification is the act of testing the logic of the model to determine that it behaves as the simulator intended. In short, it is "debugging" the computer program. During the verification process, the model may be driven by real or imaginary data, but is usually driven by simplified data so that the modeler can follow the logic of the model in detail by hand calculations. We verified the IAPS model components by using a detailed trace which printed the occurrence of each event in chronological order and the value of any variable whenever it changed. Further discussion of verification techniques can be found in standard simulation texts such as those by Shannon [SHA75], or Fishman [FIS73].

Verification is to be distinguished from validation, which is the act of comparing the model to the existing system. As a part of our IAPS simulations, we compared the IPSS output statistics from the model of the standard IBM 360/30 to data collected at the

Ft. Stewart DDC on 2 August 1979 during an actual run of the SIDPERS application system. Results are presented in Table 7-1. All times are in minutes and seconds.

The first three rows of Table 7-1 give the actual data collected at the Ft. Stewart DDC and used for validation purposes. The first row gives the elapsed time for each job step (PIA, PlB, PlC, and PlC) and the total elapsed time for all four job steps, the source of this data being the SYSLST. Since we did not model the operating system, we adjusted the elapsed times of row one by an estimated time which represented the operating system's I/O to the SYSRES pack. The amount of SYSRES I/O was again known from the SYSLST. The adjusted elapsed times, row two of Table 7-1, thus provide the primary data for validation purposes. Operator console times, row three of Table 7-1, provide a secondary source of validation data. These times were computed by actually counting the number and lengths of messages on the console log from the 2 August SIDPERS run, and by using our observation of console speed, namely 11 characters per second and approximately 1/2 second for carriage return. (IBM rates their 1052 console at 14 characters per second, but our observations and timings indicated otherwise.)

Other types of system data useful for validation purposes include CPU busy and idle times, other resource utilization, and queueing information. Due to the lack of job-step accounting we were unable to obtain any validation data other than that in Table 7-1. It is also recommended that validation data be collected from more than

Table 7-1. Validation Results

	SIDPERS Job Step				
	P1A	P1B	P1C	P1G	Total
<u>2 Aug '79</u>					
Elapsed time (minutes: second)	4:59	2:33	9:23	3:49	20:44
Elapsed time less RES I/O	4:17	2:13	8:04	3:16	17:50
Operator Console* (computed)	1:40	:45	:30	:41	3:36
<u>IPSS Model</u>					
Elapsed time	4:12	1:59	8:0	3:8	17:19
% difference	-2%	-10%	-0%	-3%	-3%
Operator Console	1:35	:54	:25	:41	3:35
% difference	-5%	+20%	-17%	+0%	-.5%
*at 11 characters/second and 1/2 second carriage return					

one run of SIDPERS, and if such data is used for model calibration purposes, that a second independent set of data be collected for validation purposes. Due to lack of time and the difficulty of obtaining such data, we were unable to obtain more than one set of data. Our identification of data sources (Table 4-1) should ease data collection in future studies.

Table 7-1 also gives IPSS output statistics of elapsed time and console times for the model of the IBM 360/30, plus percent difference between the validation data and the model data. As can be seen, overall elapsed time differed by only 3% and console time by less than 1%. Based on the limited data available to us, we accepted our model as valid.

7.2 ANALYSIS OF SIMULATION RESULTS

The main statistic of interest in our simulation study was job (and job-step) elapsed time. These results are presented in Tables 7-2 and 7-3.

Table 7-2 presents the simulation elapsed time per job step and the total elapsed time for all four job steps for four variations on the IBM 360/30 system and two variations on the Honeywell system. All of the decreases in elapsed time except for B2 over B1 are to be expected, judging by the hardware characteristics and comparisons presented in Tables 5-1 through 5-4. The decrease in elapsed time of B2 over B1 (about 1 minute, 4 seconds) is due to a different placement of certain files. In its first four job-steps, SIDPERS has 8 tape files, and models A1, A2, A3, A4 and B1 model these files

Table 7-2. Simulation Elapsed Time per Job Step
(minutes:seconds)

Experiment	SIDPERS Job Step				
	P1A	P1B	P1C	P1G	Total
A1 - Standard CS ₃	4:12	1:59	8:0	3:8	17:19
A2 - fast disk	4:12	1:53	7:30	2:45	16:20
A3 - fast console	2:38	1:05	7:36	2:27	13:46
A4 - both	2:38	1:0	7:05	2:04	12:47
B1 - DAS ₃	1:24	0:42	2:22	1:08	5:36
B2 - DAS ₃ (2 tape, 3 disk)	1:02	0:22	2:21	0:47	4:32

Table 7-3. System Comparison

Base System	Alternate System		SIDPERS Run-time on Alternate System * (hours:minutes)
	System**	Alternate System time/Base System time	
A1 (Standard CS ₃)	B1	.32	2:34
	B2	.26	2:05
	A2	.94	7:31
	A3	.80	6:24
	A4	.74	5:55
A4 (CS ₃ with fast console and 3330 disk)	B1	.44	3:31
	B2	.35	2:48
*Assumes a base system run time of 8 hours			

**
 B1 - Honeywell Level 6 Model 47 (DAS₃)
 B2 - DAS₃ with 2 tape, 3 disk units
 A2 - CS₃ upgraded by 3330 disk
 A3 - CS₃ upgraded by fast console
 A4 - CS₃ with both 3330 and fast console

as tape files. For model B2, however, the last six of the tape files are placed on disk. (The remaining two tape files are the SIDPERS input transactions.) As can be seen by examining Tables 5-1 and 5-2, the average time to transfer a block of data is faster for disk than for tape.

Two things should be kept in mind when examining Table 7-2. First, we did not model the availability of storage space. No claim is made that any configuration (especially B2) will be adequate for the storage of SIDPERS files. Second, we assumed that all tapes are premounted and that the operator takes no more than 10 seconds per job-step to respond to console messages. These two assumptions are consistent with our observations at Ft. Stewart. However, premounting of tapes may become more difficult on a faster system (e.g., B1) or impossible on a smaller system (e.g., B2 with only 2 tape units).

Keeping these limitations in mind, plus the restriction of our model to the first four jobsteps of SIDPERS, we can make a few tentative conclusions based upon Table 7-2. We see that the best upgrade of the CS₃ system, namely A4, improved performance in terms of elapsed time by approximately 25%. On the other hand, either of the two DAS₃ configurations improved performance by approximately 70% or more.

Table 7-3 presents a comparison of system A1 to its alternates, and a projection of SIDPERS run time. For example, the first four jobsteps of SIDPERS ran on system B1 in 32% of the time required on A1.

If the first four job-steps were representative of all of SIDPERS, then we would predict that an 8 hour SIDPERS run on A1, the IBM 360/30, would take 2 hours and 34 minutes on B1, the Honeywell Level 6 Model 47 (with 6 disk and 6 tape units). We emphasize that the right-hand column of Table 7-3 should not be taken as a firm prediction, but is merely for illustrative purposes. Such a prediction could only be made after modeling all or at least a substantial portion of SIDPERS. Table 7-3 also contains comparisons of the "best" upgraded version of A1, namely A4, to the two Honeywell configurations, B1 and B2.

The results in Tables 7-2 and 7-3 are illustrative of the type of results and comparisons that can be made when evaluating computer systems. Similar comparisons could be made of other quantities of interest, such as resource utilization, queueing times for resources under contention, and response time in an interactive environment.

7.3 ADDITIONAL EXPERIMENTS PERFORMED

To demonstrate the ease of model building after a library of model components is in place, we made a number of additional simulation runs.

The purpose of the first set of runs was to investigate the variability of the estimate of elapsed time due to the random elements in the model. In all experiments, random access was modeled by picking the next record to be read (or written) in a random fashion, by making use of the GGU3 random number generator, a routine in the IMSL mathematical and statistical subroutine package which is documented in [LEA73]. Another source of randomness was the random placement of files on disk when their

placement was unknown. The result of these and other uses of the random number generator is to make the estimate of elapsed time a random variable.

For experiment A1, three independent runs were made using three independent sources of random numbers. (Run 1 in each case is the run presented in Table 7-2.) The results are presented in the first 3 rows of Table 7-4. As can be seen, elapsed times for runs 1 and 3 were identical (when rounded to the nearest second), and the elapsed time for run 2 differed by only 1 second in job-step P1C. This lack of variability of the estimate of elapsed time increases our confidence in its precision. Table 7-4 also presents the results for 2 independent runs each of experiments B1 and B2. Similar conclusions can be drawn from these results.

The purpose of the second set of runs (A5, A6, A7, and A8) was to demonstrate the ease of building models from an existing library. All of the eight additional runs in Table 7-4 were made by recombining existing elements of the library, and took less than one hour to submit from an interactive terminal using the technique illustrated in Figure 3-4. All four of these models represented upgrades of the CS₃ system (A1). In experiment A5, the 2314 disk units were replaced by 3340 disks. In A6, the memory cycle time was reduced by 50% to measure the effect of doubling CPU speed. In A7, six of the eight tape files in the first four jobsteps of SIDPERS were placed on disk, so that model A7 faced the same loading as did model B2. Finally, model A8 was identical to model A4 but its loading was that of A7 and B2.

Table 7-4. Simulation Elapsed Time per Job Step

Experiment	Run	SIDPERS Job Step (Elapsed time in minutes:seconds)				
		PIA	PIB	PIC	PIG	Total
A1	1	4:12	1:59	8:0	3:08	17:19
	2	4:12	1:59	7:59	3:08	17:18
	3	4:12	1:59	8:0	3:08	17:19
B1	1	1:24	0:42	2:22	1:08	5:36
	2	1:24	0:42	2:22	1:08	5:36
B2	1	1:02	0:22	2:21	0:47	4:32
	2	1:09	0:22	2:24	0:47	4:42
A5 *		4:12	1:51	7:17	2:39	15:59
A6		3:25	1:46	5:04	2:30	12:45
A7		3:35	1:41	7:31	2:47	15:34
A8		1:57	0:40	6:35	1:41	10:53

* A5 - A1 with 3340 disk

A6 - A1 with memory cycle time reduced by 50%

A7 - A1 with 2 tape files (loading identical to that for B2)

A8 - A4 with 2 tape files

Again we emphasize that the systems modeled whose results are exhibited in Table 7-4 were chosen only to illustrate the ease of model building when using the IAPS methodology and the types of results which a valid simulation can give a decision-maker.

8. SUMMARY OF PROJECT ACTIVITIES

Three people were assigned to this project for the methodology and model building tasks. In addition, IPSS maintenance support was provided by a half-time undergraduate student. Work began on approximately 14 June 1979 and continued through 14 September 1979. The IPSS models of the IBM and Honeywell computer systems were developed, verified and validated as of 21 August 1979.

Excluding the half-time student, a total of 189 man-days were authorized for this project, of which approximately 120 were used. Table 8-1 provides a breakdown by major category. Twelve days were spent in developing the methodology and 24 in determining what validation data was available at which computer installations. This is considered to be a one-time cost. The User activities took 18 man-days, exclusive of documentation. The Modeler activities consumed 38 man-days, 25 of which were in examining SIDPERS. Fifty-six days were spent developing the IPSS code for the IAPS methodology, and 5 days were spent at the IPSS Analyst level of detail. Documentation consumed 21 days and project start-up used 5 days.

Table 8-2 compares the current research project with estimated man-day costs for several different continuing projects of similar scope. The first column gives the actual man-days for the current project, and is taken from Table 8-1. Our first projection is for a project which could essentially be a continuation of the current project. If the user wanted to run additional simulations with already existing members of the library, or wanted to consider additional minor variations on the IBM and WA systems, we project

Table 8-1. Breakdown of Project Activities

<u>Methodology</u>	
o Design of IAPS methodology	12
o Determine availability of validation data	<u>24</u>
	36
<u>User</u>	
o Determine architectures and variations to be modeled	2
o Execute IPSS models from libraries	1
o Validation	5
o Analysis and Evaluation	<u>10</u>
	18
<u>Modeler</u>	
o Characterize IBM 360 Model 30 in IPSS	4
o Characterize Honeywell Level 6 in IPSS	9
o Develop SIDPERS processing characteristics (site visit, study COBOL programs and console logs and SYSLIST and tape DITTO, encode data)	<u>25</u>
	38
<u>Simulator</u>	
o Develop IPSS routines to input application processing tables	10
o Develop IPSS routines to process application processing tables	21
o Code and verify the overall structure of the IPSS model	<u>25</u>
	56

Table 8-1 Continued.

<u>IPSS Analyst</u>	
o Study the internal logic of IPSS on a special-case basis	5
	5
Miscellaneous	
o Project start-up time	6
o Documentation	21
	27

Table 8-2. Man-Power Analysis and Projection

Activity	Current Research Project	Projections in Man-Days			
		Further CS ₃ , DAS ₃ SIDPERS Evaluation	Evaluation With All of SIDPERS	Different Hardware and Software	Operating System, CS ₃ DAS ₃
Methodology	36	0	0	0	30
User	18	10	10	15	15
Modeler	38	0	20	40	45
Simulator	56	0	0	15	50
IPSS Analyst	5	0	0	0	5
Start-Up and Documentation	27	10	10	10	25
Total Man-Days	180	20	40	80	170

a man-day cost of 10 days for running the simulations and analyzing the results, plus 10 days for start-up and documentation.

The second project we consider, of slightly greater scope, consists of comparing the CS₃ and DAS₃ systems with all of SIDPERS as the workload. Modeler activities would consist of a projected 20 days to examine the relevant COBOL application programs and to translate the sequence of I/O processing into the IAPS Application Processing Table, to collect the necessary data and to encode it into the System File Table and Application File Table; and finally to add these new members to the library. User activities would then consists of a projected 10 days for making runs and analyzing the results, plus 10 days for documentation.

The third project involves the development of the capability to model hardware other than the CS₃ and DAS₃ systems, and to model additional application software such as STANFINS. The addition of new hardware capabilities would require a projected 20 days of Modeler activities and 15 days of Simulator activity. Specific tasks to be performed would include characterization of the new hardware, data collection, coding of the data into IPSS statements, the writing of a channel program, and the addition of these new members to the library. The modeling of additional software would be a project of approximately the same scope as the second project. In summary, this third project, with a total of 80 projected man-days, would be of a scope similar to the current project, but would require 100 fewer man-days of effort because of our previous development of the IAPS methodology and the pre-existing library of model components

which represent application (I/O) processing and thus can be used with any hardware configuration or any new application software.

The fourth proposed project consists of extending the currently existing models to include operating system components. The current models do not include a representation of the operating system. New methodology would have to be developed, taking a projected 30 man-days and involving persons highly familiar with the operating systems for the IBM 360/30 (namely, DOS-E) and with that for the Honeywell system. Modeler, Simulator, and IPSS Analyst activities would require a projected 100 man-days. Specific tasks would include extensive consultation with operating system experts and data collection, plus the development of IPSS code to represent the operating system. User activities to run the model, validate it, and evaluate the output would take a projected 15 days, plus a projected 25 days to document the new members of the library and the simulations performed. At least 90 of the total projected 170 man-days would be one-time costs, after which the operating system model components would be available in the model library for future use.

Provided that an extensive library of model components were built up and maintained, future projects of the scope discussed here would tend to take less time than projected. The building and maintaining of a large and extensive model library of various hardware and software components is the key to providing timely answers to those questions which can be addressed by simulation.

9. SUMMARY AND CONCLUSIONS

SUMMARY

This project was an intensive, short-term research and development effort focused on the simulation of computer systems for the U.S. Army Computer Systems Command. Specifically, we addressed the problem of providing a model development tool which would be responsive to meeting Command simulation objectives. This required a methodology for model development, use, and analysis which would be easy-to-use, widely applicable to many types of computer systems, amenable to change, and time-efficient.

We designed, developed, implemented, and tested a methodology to meet these objectives. Our methodology, called IAPS (IPSS Application Processing System), structures the modeling process into hierarchical levels which identify specific tasks in the modeling cycle. These levels are named for the person or persons who are responsible for the activities defined within a level. A User is responsible for the overall evaluation effort. He produces an evaluation of a specific computer performance problem through (a) interactive model building in an easy question and answer format (which results in the submission of a model for computer execution) and (b) analysis of the results. The procedure for building a model uses building-block components from a model library. The role of the User presupposes that a Modeler has provided the appropriate building-blocks and has made them available for the User in the model

library. The Modeler characterizes computer hardware, the software for application processing systems, and the data files. These latter two elements are characterized in a language that we designed and implemented as part of this project. In the role of Simulator, we also wrote the program which translates these characterizations into performance statistics. The Information Processing System Simulator (IPSS) language served as our base. IPSS provides specially designed built-in hardware and software language statements which greatly facilitated our programming task. We completed the Simulator's task for a large class of application processing systems. Further effort, however, is required for modeling advanced features such as data base management systems, operating system functions, and teleprocessing.

This methodology was applied to an existing U.S. Army software system (SIDPERS) run on several IBM and Honeywell computer configurations. As Modelers, we visited an operational computer installation and collected data on a SIDPERS daily cycle. We also determined performance specifications on the IBM Model 30 computer and the Honeywell Level 6 minicomputer. This software and hardware data was encoded into IAPS source statements. Then, as Users, we built models using our interactive approach and conducted a set of experiments to analyze the performance of several architectural variations, all executing with the standard SIDPERS workload. We verified and validated our model of SIDPERS and its execution environment (an IBM 360 Model 30 computer). We then projected execution times for SIDPERS on a Honeywell Level 6 minicomputer. Our results reflect the faster CPU and peripherals of the Level 6 minicomputer.

We varied the type and speed of the peripherals on both systems to demonstrate the responsiveness capabilities inherent in our IAPS methodology. Our primary measure in evaluating these alternative configurations was total elapsed time to run the SIDPERS job. We also obtained queuing and resource utilization statistics since these are automatically generated by IPSS.

CONCLUSIONS

The objective of this project was to produce a model building methodology for simulating U.S. Army computer hardware/software systems. The project definition required a demonstration of our methodology by building models of an existing as well as a future U.S. Army computer system.

We designed our methodology based on our perception of current Army simulation needs. We implemented the methodology using the Information Processing System Simulator (IPSS), and we tested it using a subset of the programs in the SIDPERS basic cycle. Two major conclusions can be drawn from our efforts. One relates to the use of IPSS in modeling U.S. Army computer systems, and the other relates to the IAPS methodology for expressing application processing software and files. We conclude that IPSS is an appropriate tool for simulating the type of computer systems found within the U.S. Army. These systems are typified by a single processor, supporting either uniprogramming or multiprogramming, with I/O oriented COBOL file processing applications. IPSS incorporates special language features for characterizing computer hardware and files which make it especially

suited for the performance modeling and evaluation of these systems. The IPSS "service" concept helps the Simulator to produce a structured solution to complex model design problems.

The IPSS methodology and language proved to be relatively easy to learn and use. Two of the three researchers involved in this project had no prior IPSS modeling experience. With a few tutorials and IPSS models as a guide, they became productive IPSS modelers in a short period of time. The services of an IPSS expert, however, were required throughout the project.

Although IPSS is a prototype system, no IPSS source code had to be changed to generate the results produced in this report. We did identify enhancements, however, and these are detailed in the next section.

Our second major conclusion is that the IAPS methodology, and our implementation of its concepts, is an appropriate and useful method for characterizing U.S. Army computer hardware/software systems. Using IAPS, we were able to represent many types of file processing quickly and easily. In addition, the representation of computer hardware and the use of this hardware during file processing is one of the recognized advantages of IPSS.

We specifically designed IAPS with the objective of flexibility, generality, ease of use, and responsiveness. We tested and revised the methodology and the implementation during the project to more completely satisfy these objectives. We demonstrated flexibility and generality by modeling two different types of computer systems and several hardware variations. We incorporated ease of use by a library building

block approach to model synthesis and an interactive dialogue. We verified the responsiveness of the IAPS methodology by modeling some of the variations on short notice.

RECOMMENDATIONS

Our recommendations focus on three areas. First we present our recommendations for (1) further development of the IAPS methodology; (2) use of the methodology by the Computer System Command for further modeling of computer systems; (3) enhancement of the IPSS system itself.

IAPS Recommendations

Our experience as a User of the IAPS methodology suggests that it could be extended to allow a more sophisticated dialogue during model synthesis. We recommend the generation of library members from parameters input by the user. For example, the User could enter a small number of parameters for a sort operation, and the IAPS could generate the appropriate library member for this particular sort file processing. This enhancement would speed the modeling process by increasing the flexibility and generality of the library members. In addition, the interactive model synthesis could have an option such as "tutorial mode" to guide the novice model builder in great detail through every step of building a model from the model library. Such an addition to IAPS would greatly increase its ease of use and make model building a self-taught procedure.

The IAPS methodology could also be extended to include the simulation of data base management systems, operating system processing

and networks of computers. These would increase the scope of the methodology as well as the accuracy of the results obtained.

In addition, the IAPS methodology could be enhanced to allow the Modeler to represent computer hardware as he now represents computer files. This would allow greater flexibility in accommodating variations of hardware characterizations during experimentation.

Recommendations on the Use of IAPS

We recommend a continuation of the modeling effort which began with this project. In particular, we recommend modeling more of the SIDPERS basic cycle in order to further test the methodology and to verify our projections. This study should produce insights into selecting representative subsets of large systems for modeling and analysis.

We recommend the establishment of model libraries incorporating common computer architectures and software systems. This will enable the Computer Systems Command to respond quickly to future simulation needs.

We also recommend an IAPS simulation study be undertaken which involves a hardware modification. This would involve simulation and measurement of the system before and after the modification. This type of study would provide insights into the computer modeling process as well as a validation of the IAPS approach. The result would be increased confidence in the results of this type of simulation study.

IPSS Recommendations

With minor exceptions, IPSS proved to be a useful and appropriate tool for our modeling purposes. Many of our recommendations for the improvement of IPSS are already recognized and are in the process of being remedied. In particular, we make the following recommendations:

1. IPSS contains few implemented features for modeling CPU activity. Since circumstances did not permit us to model the CPU in any detail, this problem did not have a major impact upon our project, but may indeed affect any future modeling projects.
2. We were forced to rely on existing models and statement parsers to determine which options of the IPSS source code have been implemented. We were provided with a preliminary copy of a document that would remedy this situation, but its numerous errors rendered it useless. The corrected version of this document should be published, however, in the near future.
3. IPSS provides only 10 seeds to a random number generator and better random number generators are known to exist. This limits the number of independent experiments one can run to 10. We included a better random number generator and programmed a routine to accept a seed as input to the model and write out the last seed on model termination. These changes should be incorporated as a standard part of the IPSS package.
4. IPSS does not allow the modeler to save the load module and to execute the load module as a separate job (IPSS abnormally

- terminates when we tried this). As a consequence, every time an experiment is to be performed, the IPSS source language compilation and Fortran source language compilation process must occur. This consumed at least one minute CPU time for our models. We wrote special routines to bypass this problem. These routines should be incorporated as a standard part of the IPSS package.
5. We could not conveniently model concurrent activities since the IPSS automatic save/restore feature was not present in our copy of IPSS.
 6. We could not declare data sets with a BLOCK reference unit due to an error in the Fortran built-in \$CRDS routine. However, we were able to work around this error.
 7. IPSS does not automatically collect statistics on UNIT RECORD or UNSPEC type devices. CREATE DATA SET and GET ADDRESS are two very useful IPSS built-in routines that only work on disk and tape devices. We modeled the operator's console as a Tape device to easily generate the utilization statistics we wanted.
 8. A final area of possible enhancement of IPSS lies in the presentation and choice of statistical results. If desired by the user, quantities such as elapsed time should be converted from the simulation time unit (e.g. milliseconds) to hours, minutes, seconds. It also should be possible to have results tabulated and printed both cumulatively and over user specified intervals. We modeled SIDPERS at the

AD-A104 888

BATTELLE COLUMBUS LABS OH

F/6 9/2

AN IPSS-BASED MODEL-BUILDING METHODOLOGY FOR RANKING AND EVALUA--ETC(U)

OCT 79 J D BROWNSMITH, J S CARSON

UNCLASSIFIED

NL

2 OF 2

AD/A
103 488



END
DATE
FILMED
10-81
DTIC

LIST OF REFERENCES

- ADL75 Adler, M., M. Wright, B. Conway and D. Carroll. 1975. SIDPERS Computer Simulation Model. Technical Documentary Report, U.S. Army Computer Systems Command. USACSC-AT-75-05.
- BRO77 Brownsmith, J.D. and L.S. Chandler. 1977. Technical Report to the U.S. Army Computer Section Command.
- DEL77 DeLutis, T.G., J.D. Brownsmith and J.S. Chandler. 1977. An IPSS Model of SIDPERS/IDMS. Technical Report to the U.S. Army Computer Systems Command.
- DEL77 DeLutis, T.G. 1977. A Methodology for the Performance Evaluation of Information Processing Systems. Final report to National Science Foundation, OSIS GN 36622.
- DEL78a DeLutis, T.G. 1978. Information Processing System Simulator (IPSS): Syntax and Semantics, Volumes 1 and 2, working document to NSF Grant 36622.
- DEL78b DeLutis, T.G. 1978. A Simulation Language for Evaluating Information Processing Systems from a DBMS and User Perspective: Extensions to the Information Processing System Simulator. Technical Report to the U.S. Army Research Office.
- FIS73 Fishman, G.S. 1973. Concepts and Methods in Discrete Event Digital Simulation. Wiley.
- LEA73 Learmonth, G.P. and Lewis, P.A.W. 1973. Naval Postgraduate School Random Number Generator Package LLRANDOM, NPS55LW73061A, Naval Postgraduate School, Monterey, California.
- MIH76a Mihram, G.A. 1976. Four Questions Regarding the Credibility of Simulation. Modeling and Simulation 7(2): 1225-1230.
- MIH76b Mihram, G.A. 1976. Four Further Questions Regarding the Credibility of Simulation. Modeling and Simulation 7(2): 1231-1234.
- ROS78 Rose, L. 1978. Computer Systems/Database Simulation. Technical Report to the U.S. Army Computer Systems Command - AIRMICS.

- ROS79 Rose, L. 1979. IPSS: A Language and Methodology for Information Processing System Simulation. Simulation - Management Workshop, Atlanta, Georgia, p. 142-174.
- SCH77 Schaaff, H. 1977. Description of the VIABLE Transactions for the DIMUI Model. USACSC.
- SID76 SIDPERS User Manual. 1976. SIDPERS Guide for Commander and Staff. DA Pam 600-8-8.
- SID79 Standard Installation/Division Personnel System (SIDPERS) Operations and Scheduling Manual DOS version. 1979. United States Army Computer Systems Command. USACSC Manual 18-1-B-AAc. Volume III (DOS).
- SHA75 Shannon, R.E. 1975. Systems Simulation, the Art and the Science. Prentice-Hall.
- SWE76 Swenson, H. et al. 1976. Simulation of the Six Standard Army Multicommand Management Information Systems (update). Part I final report SIDPERS. SAI COMSYSTEMS, McLean, Va. Technical Report to the U.S. Army Computer Systems Command.
- VAN69 VanHorn, R. 1969. Validation. In the Design of Computer Simulation Experiments (Ed.: T.H. Naylor). Duke University Press. Durham, North Carolina.
- WHI79 Whitaker, R.M. 1979. Meeting Among Representatives of AIRMICS, Contract Researchers and QAD. Memorandum for Record. 2 July 1979.
- WHI79 White, P. 2 July 1979. Memorandum to Director, Quality Assurance, AIRMICS.

APPENDIX A

AN OVERVIEW OF THE INFORMATION PROCESSING
SYSTEM SIMULATOR (IPSS)

This Appendix highlights the IPSS methodology for characterizing salient features of information processing systems, the IPSS simulator, and the IPSS execution facility. This Appendix was extracted from previous reports prepared by Dr. L. L. Rose, Assistant Professor, The Ohio State University (ROS78, ROS79).

A.1 THE IPSS METHODOLOGY

IPSS provides a methodology which, although specific to computer systems, is general in nature, and quite flexible. It affords the user a viewpoint from which he can construct a simulation model of any computer system at any level of detail desired. This methodology separates the characterization of a complex information processing system into separate, inter-connected components. It gives structure and direction to the user, who has the difficult task of defining just what it is he wishes to model.

Figure A-1 illustrates the role of the IPSS methodology in the design and simulation of an information processing system. We observe that IPSS provides the modeler a top-down approach to the definition of models. At the top of this figure we denote the loose connection of user system knowledge into a set of data and concepts that describe the Information System. This definition may be concise and complete, showing complete knowledge of the system and processes to be modeled; it may be very vague in all respects; it may be specific with regard to certain aspects and non-specific with regard to other aspects of the information system. It is the role of the IPSS methodology to enable the modeler, who possesses varying degrees of information about the information system, to construct a model at appropriate levels of detail to satisfy his modeling needs.

The IPSS methodological view is to characterize any information

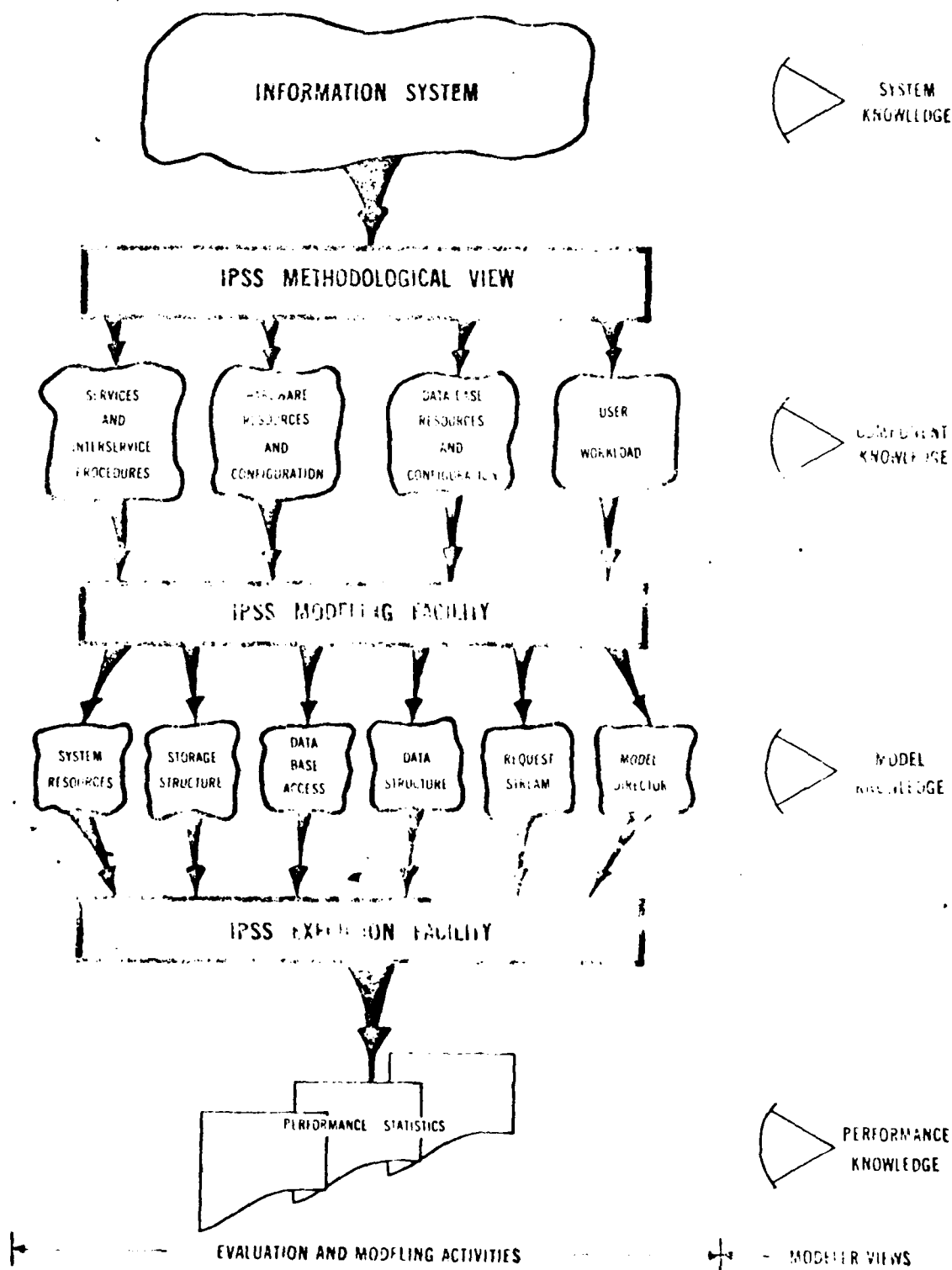


Figure A-1. The IPSS Methodology

processing system as a collection of four discrete but interfacing components. As illustrated in Figure A-1 these components are:

- 1) services and inter-service procedures, 2) hardware resources and configurations, 3) data base resources and configuration, and 4) user workload. These four component definitions are sufficient to characterize any information processing system; in particular, computer-based information systems or manual systems can be described.

Services and Inter-Service Procedures

The identification and definition of services and inter-service procedures is an important IPSS contribution, and separates its methodology (and subsequent modeling activities) from other systems such as DIMUI and CASE. A service procedure defines a task - manual or automatic - associating all related actions and times to complete the task. In a computer-based system, this component corresponds to the definition of all system software facilities, to include user application programs, the operating system, and the data management system. Service definitions, of course, are constrained to the level of detail required by the modeler or to the level of knowledge of the modeler. This is true of all four component definitions, and forces the modeler to realize the level of detail appropriate, and to obtain additional information, if required, to properly define each component. Note that no computer programming is being performed at this time; we are structuring the model to be defined and isolating user information into the

appropriate sets of component knowledge. In any computer programming activity, too much emphasis cannot be placed on structuring the prototype, for correct and appropriate structure can be followed by easy implementation which, by design, should reflect the needs of the modeler.

Hardware Resources and Configuration

The hardware resources and configuration component directly reflects the hardware system to be modeled. This component defines the CPU, primary storage, tapes, discs, drums, printers, terminals, channel controllers, etc., and all hardware interconnections. Again, the level of detail required is that appropriate to the goals of the modeling activity.

Database Resources and Configuration

The database resources and configuration component defines the logical database of the system to be modeled, to include schemas, file characteristics, database access capabilities, and user data access and data manipulation facilities. This component can reflect a current system with normal non-integrated file management or a future system with fully integrated data management capabilities.

User Workload

Last, but certainly of great importance, is the user workload component. It is here that one characterizes the workload to be placed on the simulated system, to include workload description, timing of inputs, files referenced, etc. This completes the structuring of the user's knowledge of the information system and

can be defined functionally or statistically.

A global view of the resultant component is as follows:
work (input) to the information processing systems emanates from the user workload and requires certain services. These services may require other services (inter-service procedures) to perform the work required. Whenever database accesses are required, the database resources and configuration component defines and simulates logical data flow while the hardware resources and configuration component simulates the resultant physical data flow. This is the user's view of the information flow process at the conceptual level, structured into components by the IPSS methodology.

A.2 THE IPSS MODELING FACILITY

Given the user's component knowledge as structured by the IPSS methodology, this is transformed by the modeler into model knowledge using the IPSS modeling facility. This portion of IPSS also provides structure and modularity to the model definition, but at a realizable level, as opposed to the conceptual level of component knowledge. The result of this transformation from component to model knowledge is an IPSS-defined simulation model that can be executed by the IPSS execution facility.

There are six model components which comprise the resultant defined model. Given the separation of user knowledge into the four conceptual components defined previously, it is a straightforward task, conceptually, to define the six IPSS model components

which describe system resource, storage structure, database access, data structure, request stream and model director. To actually implement these modules represents a non-trivial, sophisticated effort that requires not only a good understanding of the system to be modeled, but also a complete understanding of how to effectively simulate all of the concepts and interactions of the process to be modeled.

IPSS provides a general simulation language and host environment to ease this task for the modeler. The Model Director is supplied for the user, and, in effect, directs the simulation defined by the other five model components. It handles the time clock, and the events queues, and all arrivals and departures from the system during model simulation. CASE and DIMUI effectively pre-define the entire simulation model (especially the system resources model component). This results in much less understanding about the model; it is the IPSS premise that a modeler cannot effectively use a simulation model that he does not understand.

As a result, IPSS offers a set of language constructs so that the user can, with relative ease, define all important aspects of the simulated activity. Using the IPSS statements, and any additional FORTRAN the user may desire, a FORTRAN model is output from the IPSS translator which can be executed to produce statistics. Additional FORTRAN statements are utilized by the modeler to either add statistics unavailable from IPSS or to model concepts not realized by the IPSS language constructs. In most cases, little additional FORTRAN is required as

IPSS provides a rich set of language constructs with associated statistical capabilities.

The top-down, modular approach provided by the IPSS enables the user to define, using IPSS/FORTRAN statements, five separate model components to characterize the system to be modeled. These are summarized below:

1. System Resources - Contains definitions for all information system resources (hardware and software) and all system tasks (application and operating system). This component forms the basic discrete event digital simulator for the information systems model under investigation. Included in the SYSTEM (system resources component) is the IPSS supplied clockwork mechanism to schedule and control simulated events and to determine when the simulation is to terminate. The clockwork logic is based on the next most immediate event philosophy for controlling discrete event digital simulations.

IPSS statements which ease the modeler's task of defining all of the system resources pertinent to the simulation desired include: Access Mechanism, Area, Buffer Pool, Central Processor, Control Unit, Data Channel, Data Set, Device, Endo Service, Exo Service, I/O Processor, Main Storage, Path, Procedure, Queue, Reference, Semaphore, Task, and Volume statements.

2. Storage Structure - Describes an information system's physical data base storage structure and its space management policies. The STORE (storage structure) component interfaces with the SYSTEM component in three ways. First, it references

SYSTEM to obtain Device and Volume facility definitions. Second, it supplies SYSTEM with Data Set facility definitions. Third, it translates secondary storage references specified as a displacement within a data set's logical address space into physical addresses within the secondary storage address space. Prior to a simulation, associations must be specified for the Data set, Organization Method, Device and Volume facilities. A STORE Organization Method facility can be associated with a multiple number of SYSTEM Data Set facilities. The opposite is true for the Device and Volume facilities. STORE Organization Method facilities are the templates from which the equated SYSTEM Data Set facilities derive their definitions during a simulation. The transfer of definitions between components is accomplished via the execution of the CREATE DATA SET Statement. The space management descriptions in STORE are used to calculate secondary storage addresses dynamically during a simulation based on facility definitions specified in each component and on the changes of these facilities during the course of the simulation.

IPSS statements provided to help the modeler define the Storage Structure Component include: Area, Segment, Organization Method, Extent, Record Type, Device, Procedure, Reference, and Volume.

3. Request Stream - Characterizes the information system's service request stream. It is responsible for the generation of all exogenous events for a model. Whereas SYSTEM contains facilities which characterize the processing requirements for each service offered by an information system, the request stream component (REQUEST)

defines the arrival of request for these services. IPSS converts these times into a composite arrival time stream.

The modeler thus defines exogeneous events, and IPSS eases this task by offering the Exogenous Event statement and the Procedure Statement should the modeler desire to define inter-arrival times functionally.

4. Data Base Access - Contains the definitions of all the resources required by the DBMS. These include the hardware resources of buffers and user work areas as well as application programs and DBMS software. All DBMS related entity-type facilities are defined within the component. The Data Base Access Component (ACCESS) is similar to the SYSTEM components in that it contains its own simulation clockwork mechanism similar in purpose to the one belonging to the REQUEST component.

IPSS statements particular to the Data Base Access Component include: DML Service, Realm, Schema, Record Origin, Semaphore, Task, and Queue.

5. Data Base Structure - Provides the modeler with a set of facilities which allows the definition of logical data structures and the characterization of relationships among them. This can be applied to a variety of DBMS architectures and application environments. The Data Base Structure component (STRUCTURE) permits the modeler to investigate the effects on system behavior caused by alternate set, record type, and access path definitions. The definitional facilities provided allow the modeler to investigate a wide spectrum of logical data structure organizations and allocation policies.

Within the Data Base Structure Component are IPSS statements to enable the modeler to define the following important database constructs:

Realm, Schema, Extent, Record Type, and Set.

A.3 THE IPSS EXECUTION FACILITY

The six IPSS model components discussed in the previous section (MODEL being pre-defined while SYSTEM, STORAGE, REQUEST, ACCESS, and STRUCTURE are user-defined with the aid of IPSS language constructs) comprise the input to the IPSS Execution Facility. It should be understood, however, that this six-component model definition serves not only as necessary input to the IPSS Execution Facility. Of at least equal importance is the fact that the user has now created a documented, readable, understandable definition of the system to be modeled. The fact that this model is explicitly defined at user-determined levels of detail for each model component means that we have a hard copy description of exactly what the modeler wishes to simulate. No implicit assumptions (such as are contained in CASE and DIMUI) exist; hence user verification of the model can be accomplished much more effectively, and the entire modeling effort is at the level of detail desired by the modeler.

The IPSS execution facility carries out the simulation as defined by the six IPSS model components. This execution requires translation of IPSS statements into FORTRAN, link-editing of all required object modules, saving certain user-requested object/source modules in the IPSS library, and executing the resultant load module. Were the user required to define to the computer this multi-step job, a great deal of JCL (machine-dependent job control language) would be

necessary. In fact, both CASE and DIMUI require the user to create his own multi-step jobs, a non-trivial, machine-dependent task. The IPSS philosophy is to remove the tedium and complexity of JCL from the user; in fact, the user specifies no JCL whatsoever to execute an IPSS model. Thus IPSS must contain, within its own code, this JCL. We find this within the IPSS Nucleus, which is written in Assembler language. Hence we find that the IPSS is not completely portable, but only the Nucleus must be re-written to enable execution on another dissimilar machine.

A.4 THE IPSS STATISTICS

IPSS provides a modeler with a number of statistics concerning the behavior of modeler defined entities and IPSS supplied built-in information system services. Many output statistics are provided by IPSS automatically; others can be generated by the modeler's use of IPSS commands to start/end data collection on queues, facilities, services, etc. The IPSS-defined (automatic or modeler invoked) output statistics fall into eight general categories:

1. Operational Statistics,
2. Request Stream Statistics,
3. I/O Activity
4. Queueing Statistics,
5. Utilization Statistics,
6. Wait Statistics,
7. Service Statistics, and
8. Task/Activity Statistics.

Additionally, the modeler can employ the complete facilities of the FORTRAN language to develop his own statistics. Statistics are printed automatically at the conclusion of each model simulation unless explicitly inhibited.

APPENDIX B

IAPS SOURCE CODE

This Appendix contains examples of IPSS source code. Specifically, it contains a complete listing of the IPSS System Resources component for the IBM 360/30 (and all variants considered in this project). The System Resources component gives specifications and characteristics of all hardware components in the model. Following this are three examples of IPSS Services, which are used to represent software and application program I/O and CPU processing.

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80
 INPUT CARD IMAGE

DEFINE SYSTEM RESOURCES: NAME = SID360M10.

COMPILE = YES.

DISPOSITION = KEEP;

PROCEDURE: NAME = -MENT, TYPE = SUBROUTINE:

C MODEL OF - USACSC STOPPERS PROCESSING SYSTEM

C FOR - U.S. ARMY COMPUTER SYSTEMS COMMAND

C WRITTEN BY - JOSEPH D. BROWNSMITH, JOHN S. CARSON II, AND

C WILLIAM HOCHSTETTLER III

C DATE - JULY-AUGUST 1979

C HARDWARE - IBM 360 MODEL 30

C NOTE - THIS MODEL IS WRITTEN IN THE IPSS LANGUAGE. IPSS

C IS THE INFORMATION PROCESSING SYSTEM SIMULATOR

C THIS MODEL CONSISTS OF THE FOLLOWING COMPONENTS -

C REF SERVICE/PROCEDURE APPROX INVOICES

C # NAME PAGE CALLS DESCRIPTION

C --- -----

C 1 SYSTEM RESOURCES (01) - COMPONENT DEFINITION 00000190

C 2 COMMENT (PROC) (01) - COMMENTS 00000210

C 3 INIT (FXS) (07) 13.12 RD SYS FILE (FV1) 00000220

C 4 START (FXS) (09) 5 START APPL PROCESSING 00000230

C 5 APPL (FNS) (11) 14.6.9 APPL PROCESSING 00000240

.....15.....20.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO ALT INPUT
LEV SEQ-NO MEMBER
REF-NO

1 1

2 2

3 3

4 4

5 5

6 6

7 7

8 8

9 9

10 10

11 11

12 12

13 13

14 14

15 15

16 16

17 17

18 18

19 19

20 20

21 21

22 22

23 23

24 24

Figure B-1. An Example of IAPS Source Code

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80	INPUT CARD IMAGE	DATE ... 08/28/79	INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
C 6	INTF (FNS)	(23)	7	SYSTEM INTERFACE	00000250
C 7	FMAP (FNS)	(24)	11	APPL -> FMS MAPPING	00000260
C 8	USER (FNS)	(29)	-	USER PROC EXIT	00000270
C 9	BUFMR (FNS)	(30)	9	FMS BUFFER MGM	00000280
C 10	GADRU (FNS)	(38)	-	GET DATA SET ADDRESS	00000290
C 11	CHPGM (FNS)	(57)	10	FMS CHANNEL PGM	00000300
C 12	BUFST (FNS)	(63)	-	PRINT BUFFER STATS	00000310
C					00000320
C 13	RSUF (PRDC)	(69)	-	READ APPL PHOC TBL S	00000330
C 14	ROGP (PRDC)	(84)	16	READ EXEC GROUP	00000340
C 15	IDJC (PRDC)	(96)	-	READ I/O DEFINITION	00000350
C 16	DDC (PRDC)	(102)	-	PROCESS DELAY DEF	00000360
C 17	STORAGE STRUCTURE	(105)	-	COMPONENT DEFINITION	00000370
C 18	DSVAR (FAC)	(105)	-	DISK AREA FACILITY	00000380
C 19	DDSI (FAC)	(107)	-	DISK DATA SET FAC	00000390
C 20	TAPAR (FAC)	(108)	-	TAPE AREA FACILITY	00000400
C 21	TDSI (FAC)	(110)	-	TAPE DATA SET FAC	00000410
C 22	FXD EVENT STREAM	(113)	-	COMPONENT DEFINITION	00000420
C 23	FV1 (FXD EVENT)	(113)	-	GET SYS FILE INPUTS	00000430
C 24	FV2 (FXD EVENT)	(114)	-	START APPL PROCESSING	00000440
C 25	MODEL	(115)	-	COMPONENT DEFINITION	00000450
C					00000460
C					00000470
C					00000480
.....10.....20.....30.....40.....50.....60.....70.....80	RETURN				

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80

END: PROCEDURE:

CENTRAL PROCESSOR: ID=CPU:

MAIN STORAGE: ID=MSTCR,

SIZE = 256000:

INPUT OUTPUT PROCESSOR: ID=(SELECT.2),

MAX TRANSFER RATE = 800000:

INPUT OUTPUT PROCESSOR: ID=MUX,

MAX TRANSFER RATE = 200000:

CENTRAL UNIT: ID = CU2314,

MAX TRANSFER RATE = 312000:

ACCESS MECHANISM: ID = (DK2314.8),

VOLUME = VL2316(1.8),

DEVICE = DV2314:

VOLUME: ID = (VL2316.8),

DEVICE = DV2314:

.....10.....20.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO	ALT LEV	INPUT SEQ-NO	MEMBER REF-NO
49			45
50			
51			
52			46
53			
54			47
55			48
56			
57			49
58			50
59			
60			51
61			52
62			
63			53
64			54
65			
66			55
67			56
68			57
69			
70			58
71			59
72			

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80

DEVICE: ID = DV3314,

TYPE = DASD,

BLOCK SIZE = (VARIABLE, MAX(TRACK CAPACITY)),

CYLINDERS = 200,

TRACKS PER CYLINDER = 20,

TRACK CAPACITY = (7296, CHARACTERS),

ROTATIONAL SPEED = 2400,

TRANSFER RATE = 312000,

SPACE OVERHEAD = 0,

CYLINDER ACCESS = PROC(CYLACC):

ACCESS MECHANISM: ID = (DK3330,8),

VOLUME = VL3336(1,8),

DEVICE = DV3330:

VOLUME: ID = (VL3336,8),

DEVICE = DV3330:

DEVICE: ID = DV3330,

TYPE = DASD,

BLOCK SIZE = (VARIABLE, MAX(TRACK CAPACITY)),

CYLINDERS = 400,

TRACKS PER CYLINDER = 19,

TRACK CAPACITY = (13030, CHARACTERS),

.....10.....20.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
73		60
74		61
75		62
76		63
77		64
78		65
79		66
80		67
81		68
82		69
83		
84		70
85		71
86		72
87		
88		73
89		74
90		
91		75
92		76
93		77
94		78
95		79
96		80

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STPFAM LISTING

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80

ROTATIONAL SPEED = 3600.
TRANSFER RATE = 806000.
SPACE OVERHEAD = 0.
CYLINDER ACCESS = PROC(CYL333):

CCNTRCL UNIT: ID = CU2804.
MAX TRANSFER RATE = 120000:

ACCESS MECHANISM: ID = (TP2401.6).
VOLUME = VL2401(1.6).
DEVICE = DV2401:

VOLUME: ID = (VL2401.6).
DEVICE = DV2401:

DEVICE: ID = DV2401.
TYPE = TAPE.
DENSITY = 1600.
SPEED = 75.
IRG = .6.

START STOP TIME = 13.
BLOCK SIZE = (VARIABLE, MAX(32767)).
FORWARD PHASE LENGTH = 3.
PF*IND RATE = 750:

.....10.....20.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
97		81
98		82
99		83
100		84
101		
102		85
103		86
104		
105		87
106		88
107		89
108		
109		90
110		91
111		
112		92
113		93
114		94
115		95
116		96
117		97
118		98
119		99
120		100

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

PAGE 6

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80

CCNTRCL UNIT: ID = CU2821.

MAX TRANSFER RATE = 200000;

ACCESS MECHANISM: ID = CR2540.

DEVICE = DV254R;

DEVICE: ID = DV254R.

TYPE = UR.

CYCLE TIME = 60.

BLOCKSIZE = (VARIABLE, MAX(80)).

MODE = SYNCHRONOUS.

BUFFERED = YES;

ACCESS MECHANISM: ID = CR2540.

DEVICE = DV254P;

DEVICE: ID = DV254P.

TYPE = UR.

CYCLE TIME = 200.

BLOCKSIZE = (VARIABLE, MAX(80)).

MODE = SYNCHRONOUS.

BUFFERED = YES;

.....10.....20.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
121		
122		101
123		102
124		
125		103
126		104
127		
128		105
129		106
130		107
131		108
132		109
133		110
134		
135		111
136		112
137		
138		113
139		114
140		115
141		116
142		117
143		118
144		

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80
 INPUT CARD IMAGE

ACCESS MECHANISM: ID = PRI403.

DEVICE = DV1403:

DEVICE: ID = DV1403.

TYPE = UR.

BLOCKSIZE = (VARIABLE, MAX(133)).

MODE = SYNCHRONOUS.

CYCLE TIME = 54.5:

VOLUME: ID=FCONV, DEVICE=DV7100:

ACCESS MECHANISM: ID = CONSOL.

VOLUME = FCONV.

DEVICE = DV7100:

DEVICE: ID = DV7100.

TYPE = TAPE.

BLOCKSIZE = (VARIABLE, MAX(133)).

DENSITY = 8.

SPEED = 960.

LOG = .6.

START STOP TIME = 1.

FORWARD FRASE LENGTH = 1.

REWIND RATE = 150:

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
145		119
146		120
147		
148		121
149		122
150		123
151		124
152		125
153		
154		126
155		127
156		128
157		129
158		
159		130
160		131
161		132
162		133
163		134
164		135
165		136
166		137
167		138
168		

.....10.....20.....30.....40.....50.....60.....70.....80

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
VOLUME: ID=CCNV, DEVICE=DV1052:
ACCESS MECHANISM: ID = CNI052,
      VOLUME = CCNV,
      DEVICE = DV1052:

DEVICE: ID = DV1052,
      TYPE = TAPE,
      BLOCKSIZE = (VARIABLE, MAX(133)),
      DENSITY = 8,
      SPEED = 11,
      IRG = .6,
      START STOP TIME = 500.,
      FORWARD ERASE LENGTH = 3,
      REWIND RATE = 350:

DATA SET: ID = DDS1:
DATA SET: ID = TDS1:
DATA SET: ID = CDS1:
PROCEDURE: NAME = CFUTME,
      TYPE = SUBROUTINE,
      PARAMETER LIST = (MCTME):

REAL MCTME
C *** IBM 360 MODEL 30 RETURN MEMORY CYCLE TIME IN MICRO SECONDS.
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT. INPUT LEV SEQ-NO	MEMBER REF-NO
169		139
170		140
171		141
172		142
173		143
174		144
175		145
176		146
177		147
178		148
179		149
180		150
181		151
182		152
183		153
184		154
185		155
186		156
187		157
188		158
189		159
190		
191		
192		

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80

MCTMF = 1.5

RETURN

END: PROCEDURE;

PROCEDURE: NAME = CYLACC.

TYPE = SUBROUTINE;

REAL CYLS, \$PWLIN

REAL POINTS(4,2) /25.. 65.. 75.. 135.. 0.. 20.. 80.. 200./

C

CYLS = IABS(SYSCOM(2) - SYSCOM(1))

C

CALL \$SRL1(\$PWLIN(POINTS,4,CYLS), SYSCOM(3))

RETURN

END: PROCEDURE;

PROCEDURE: NAME = CYL133.

TYPE = SUBROUTINE;

REAL CYLS, \$PWLIN

REAL POINTS(2,2) /10.. 55.. 0.0, 434./

C

.....1.....2.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
193		160
194		161
195		162
196		
197		163
198		164
199		
200		166
201		167
202		
203		169
204		170
205		171
206		172
207		173
208		174
209		
210		175
211		176
212		
213		178
214		179
215		
216		181

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
CYLS = IABS(SYSCOM(2)) - SYSCOM(1))
C
CALL $SRL($PMLIN(PDINTS,2,CYLS), SYSCOM(3))
RETURN
C*****
END: PROCEDURE;

EXG SFRVICF: ID=INIT, NAME=INIT$;
COMMON /BP/ BPDDL(5,50,4)
INTEGER I,BPCL,J,K,PETCU,PETCS

END: DECLARATIONS;
END: INITIALIZATION;

C
C *** GET FIOS PARAMETER AND FT01,FT02 FILE TABLES
C
CTRACEWRITE(4,5)
5 FORMAT(' ..**MSG INIT BEFORE CALL $SUF')
CALL $SUF(RFTCU,PETCS)
CTRACEWRITE(4,15) PETCU,PETCS
15 FORMAT(' ..**MSG INIT AFTER CALL $SUF-PETCUPCS- ',
1 2(I2,1X))
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT LEV	INPUT SEQ-NO	MEMBER REF-NO
217			182
218			183
219			184
220			185
221			186
222			187
223			
224			
225			
226			188
227			189
228			190
229			
230			192
231			193
232			194
233			195
234			196
235			197
236			198
237			199
238			200
239			201
240			202

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

PAGE 11

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
C
  DC 20 I = 1.5
    DO 20 J = 1.50
      DO 20 K = 1.4
        20 BPOOL(I,J,K) = 0
      IF(PETCU .NE. 0) GO TO 100
    C IF(PFTCS .NE. 0) GO TO 100
  C *** CREATE ALL DATA SETS
  C
    CREATE DATASET: DATASET = D0S1, INDEX=1:
    ALLOCATE DATASET EXTENT: DATASET = D0S1, EXTENT = 1:
  C
    CREATE DATASET: DATASET = T0S1, INDEX = 1:
    ALLOCATE DATA SET EXTENT: DATASET = T0S1, EXTENT = 1:
  C
    CTRACEWRITE(4,25)
    25 FORMAT(' ',**MSG INIT AFTER FILE CREAT(S*))
    CREATE DATASET: DATASET = C0S1, INDEX = 1:
    ALLOCATE DATA SET EXTENT: DATASET = C0S1, EXTENT = 1:
  C
    100 CONTINUE
  C *****
    END: EXO SERVICE:
  C
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
241		203
242		204
243		205
244		206
245		207
246		208
247		209
248		210
249		211
250		212
251		213
252		214
253		215
254		216
255		217
256		218
257		219
258		220
259		221
260		222
261		223
262		224
263		
264		

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

PAGE 12

DATE ... 08/28/79

INPUT CARD IMAGE
.....30.....40.....50.....60.....70.....80

END SERVICE: ID = START, NAME = STARTX,

SAVE AREA SIZE = 30;

C

INTEGER I,ISEED,KSEED,NAPS, NSFL, NUFL, NUM

EQUIVALENCE (I,\$SAVE(1)),(NUM,\$SAVE(2))

COMMON /USERP/ NAPS, NUFL, NSFL

COMMON /SEED/ ISEED

C

END: DECLARATIONS:

END: INITIALIZATION:

C

SET: FACILITY = START;

NUM = NAPS

KSEED = ISEED

I = 0

5 I = I + 1

CTRACWRITE(4,12) I,NUM

12 FORMAT(' ',1000MSG START -I,NUM- ',2(16,2X))

IF (I .GT. NUM) GO TO 20

INVOKE: SERVICE = APPLX,

PARAMETER LIST = (1);

GO TO 5

20 CONTINUE

.....10.....20.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
265		225
266		226
267		227
268		228
269		229
270		230
271		231
272		232
273		233
274		234
275		235
276		236
277		237
278		238
279		
280		240
281		241
282		242
283		243
284		244
285		245
286		246
287		247
288		248

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
      INPUT CARD IMAGE
CTPACFWRITE(4,14)
      14 FORMAT(' ',*MSG START AFTER 20 AND BEFORE WAIT APPL')
      WAIT RETURN: SERVICE = APPL;
      INVOKE: SERVICE = BUFSTX;
      WAIT RETURN: SERVICE = QUFST;
CTPACFWRITE(4,24) NUM
      24 FORMAT(' ',*MSG START AFTER WAIT QUFST -NUM- ',16)
      IF (NUM .LE. 1) GO TO 90
      NUM = NUM - 1
      GO TO 20
90 CONTINUE
      #PITF(6,2) KSFED, ISFED
      2 FORMAT(//.20X, 'BEGINNING SEED FOR P.N.G. - ',110,
      +      /,20X, ' ENDING SEED FOR R.N.G. - ',110)
      RELEASE: FACILITY = START;
C *****
      END: EXD SERVICE;

      ENDG SERVICE: ID = APPL,
      NAME = APPLX,
      SAVE AREA SIZE = 30,
      PARAMETER LIST = (APPL);
C
.....17.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
289		249
290		250
291		251
292		252
293		253
294		254
295		255
296		256
297		257
298		258
299		259
300		260
301		261
302		262
303		263
304		264
305		265
306		
307		
308		266
309		267
310		268
311		269
312		270

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
INPUT CARD IMAGE
C *** THIS SERVICE REPRESENTS APPLICATION PROCESSING.  EXECUTION
C *** GROUPS ARE READ IN (ONE AT A TIME) AND A SEQUENCE OF READS AND
C *** WRITES IS ISSUED TO THE NEXT (LOWER) SERVICE.
C
C THIS SERVICE
C CALLS RDPG TO READ ALL PROCESSING GROUPS OF
C      OF AN EXECUTION GROUP, AND
C CALLS EXGP TO PERFORM THE I/O FOR EACH EXECUTION GROUP
C
C APPLICATION PROCESSING TABLE*
C NUMERIC I/O TABLE - NIOT
C (APS.I.1)
C (APS.I.2)
C (APS.I.3) - INDEX OF FILE IN FILET
C (APS.I.4) - POINTER TO SYSFT
C (APS.I.5) - NUMBER OF APPLICATION RECORDS READ OR WRITTEN
C (APS.I.6)
C
C INTEGER APS,APST(5),APSK,IUPT(20),RTIC,UNIT
C END: DECLARATIONS:
C APS = APSK
C CFT PROCESSING GROUP FROM EACH APPLICATION
C

```

```

.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT LFV	INPUT SEQ-NO	MEMBER REF-NO
313			271
314			272
315			273
316			274
317			275
318			276
319			277
320			278
321			279
322			280
323			281
324			282
325			283
326			284
327			285
328			286
329			287
330			288
331			289
332			290
333			291
334			292
335			293
336			294

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

PAGE 15

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
END: INITIALIZATION:
SEIZE: FACILITY = APPL:
WRITE(6,7) APS
7 FORMAT(1H1,15X,'I/O PROCESSING TABLE FOR APPLICATION ',12)
UNIT = 0 + APS
3 READ(UNIT,1,END=4) (IDPT(J), J=1,20)
1 FORMAT(20A4)
WRITE(6,2) (ICPT(J), J=1,20)
2 FORMAT(10X,20A4)
GC TO 3
4 REWIND UNIT
C BEGIN EXECUTION GROUP PROCESSING HERE
5 CONTINUE
CTRACEWRITE(4,6) APS
6 FORMAT(' *****56 APPL AT BEGINNING -APS- ',(6)
CALL FDCP(APS,RETC)
APST(APS) = RETC
C RETC = 1 OK
C 2 FCF - NO DATA TO BE PROCESSED HERE
C 3 NO DATA - DON'T PROCESS IT HERE
C 4 TOO MUCH DATA - DON'T PROCESS IT MORE
IF(RETCLF=1) GO TO 8
IF(RETCLF=2) GO TO 999
C PAID DATA OF TOO MUCH DATA - GET NEXT FCF GROUP
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
337		295
338		296
339		297
340		298
341		299
342		300
343		301
344		302
345		303
346		304
347		305
348		306
349		307
350		308
351		309
352		310
353		311
354		312
355		313
356		314
357		315
358		316
359		317
360		318

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
      INPUT CAPD IMAGE
      GC TO 5
      C PROCESS EACH PROCESSING GROUP OF THIS EXECUTION GROUP
      8 CONTINUE
      INVOKE: SERVICE = EXGPX.
      PARAMETER LIST = (APX);
      WAIT RETURN: SERVICE = EXGPX;
      C PRINT STATISTICS AFTER COMPLETION OF EACH EXECUTION GROUP
      CALL $SNAP
      TRACEWRITE(4,2)

      GC TO 5
      900 CONTINUE
      RELEASE: FACILITY = APPL;

      C *****
      C *****
      END: END SERVICE;

      ENDO SERVICE: IC = FXGPX.
      NAME = EXGPX.
      SAVE AREA SIZE = 30.
      PARAMETER LIST = (APX);
      COMMON /USERP/ NAPS, NUFL, NSFL

      .....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
361		319
362		320
363		321
364		322
365		323
366		324
367		325
368		326
369		327
370		
371		329
372		330
373		331
374		332
375		333
376		334
377		
378		
379		
380		335
381		336
382		337
383		338
384		339

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
C 3 FACTYP,FDD,FDL,FEDD,FID,I,LREC,LRECL, 00004520 1537 1474
C 4 MDS,NBPR,NBPRU,NEXT,NF,NSEG,PREC,PRECL,REC, 00004530 1538 1475
C 5 RPREC,RUTYPE,S,SDD,ST,TC,TPC,TRK,VOL, 00004540 1539 1476
C 6 FMPT,ISFT,LRECLX,NRECS,NEXT,PTEXT,PDSX,UFN0, 00004550 1540 1477
C 7 VOLNUM,VOL TYP, 00004560 1541 1478
C 8 ESIZE,NRU,PROA,PRDE,PRDS,PRDV,PRDVC,SDV, 00004570 1542 1479
C 9 SSIZE,T,X) 00004580 1543 1480
C AT 10 00004590 1544 1481
C TRACE ON 00004600 1545 1482
C ***** 00004610 1546 1483
C ***** 00004620 1547 1484
C ***** 00004630 1548
C ***** 00004640 1549
C ***** 00004650 1550
C ***** 00000010 1551 1485
C ***** 00000020 1552 1486
C ***** 00000030 1553 1487
C ***** 00000040 1554 1488
C ***** 00000050 1555
C ***** 00000060 1556 1490
C ***** 00000070 1557 1491
C ***** 00000080 1558 1492
C ***** 00000090 1559 1493
C ***** 00000100 1560 1494

```

END: PROCEDURE;

END SERVICE: ID=CHPGM.

NAME = CHPGMX.

SAVE AREA SIZE = 30.

PARAMETER LIST = (DSX,FX,LRECLX,ISFTX,RDWTX):

```

INTEGER AN,AVAIL,CHSTAT(2),CYL,DS,DSX,DVTYPE,F,FX
INTEGER IODATA,INUSE,ISFT,ISFTX,LRECL,LRECLX,PARM,PRECL
INTEGER RDWT,RDWTX,SELSW,VOL
DATA AVAIL/C/, INUSE/1/, CHSTAT/0.0/
DIMENSION PARM(5),IODATA(15)

```

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

PAGE 66

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
EQUIVALENCE (PARM(1),DS), (PARM(2),F), (PARM(3),LREC),
+ (PARM(4),ISFT), (PARM(5),RDWT),
+ (VOL,IODATA(2)), (AM,IODATA(3)), (DVTYPE,IODATA(5)),
+ (CFEV,IODATA(4)),
+ (CYL,IODATA(6)), (PRECL,IODATA(11)),
+ (PARM,$SAVE(16)), (IODATA,$SAVE)
C
C      END: DECLARATIONS:
C
C *** TRANSFER PARAMETERS TO REGULAR FORTRAN VARIABLES
C
DS = DSX
F = FX
LREC = LRECX
ISFT = ISFTX
RDWT = RDWTX
C
C      END: INITIALIZATION:
C      SET7E: FACILITY = CHPGM:
C
C *** GET THE PHYSICAL ADDRESS OF THE LOGICAL RECORD TO BE USED
C
CTRACWRITE(4,14) DS,F,LREC,ISFT
14 FORMAT(' *****MSG CHPGM BEFORE GADRU -DS,F,LREC,ISFT-',
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
1561		1495
1562		1496
1563		1497
1564		1498
1565		1499
1566		1500
1567		1501
1568		1502
1569		1503
1570		1504
1571		1505
1572		1506
1573		1507
1574		1508
1575		1509
1576		1510
1577		1511
1578		1512
1579		1513
1580		1514
1581		1515
1582		1516
1583		1517
1584		1518

INFORMATION PROCESSING SYSTEM SIMULATOR

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

INPUT CARD IMAGE
.....10.....20.....30.....40.....50.....60.....70.....80

```

1      4(15,2X))
      CALL GADRU(DS,F,LREC,ISFT,IODATA)
CTRACEWRITE(4,10) IODATA
      IN FORMAT(' ',**MSG CHPGM AFTER GADRU - IODATA-'',
1      15(15,1X))
C
C *** ACQUIRE THE FACILITIES TO EFFECT THE DATA TRANSFER. FIRST
C *** DETERMINE IF I/O IS TAPE OR DISK
C
C      IF (DTYPE .NE. 1) GO TO 200
C
C *** DISK I/O, ACQUIRE FACILITIES IN REVERSE ORDER
C
C      CUFUE: FACILITY = AM;
      WAIT FACILITY: FACILITY = AM,
          STATUS = AVAIL;
      DEPART CUFUE: FACILITY = AM;
      SET STATUS: FACILITY = AM,
          STATUS = INUSE;
      SPUFE: FACILITY = AM;
C
      CUFUE: FACILITY = CU2314;
      WAIT FACILITY: FACILITY = CU2314,
          STATUS = AVAIL;

```

.....10.....20.....30.....40.....50.....60.....70.....80

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
1585		1519
1586		1520
1587		1521
1588		1522
1589		1523
1590		1524
1591		1525
1592		1526
1593		1527
1594		1528
1595		1529
1596		1530
1597		1531
1598		1532
1599		1533
1600		1534
1601		1535
1602		1536
1603		1537
1604		1538
1605		1539
1606		1540
1607		1541
1608		1542

INFORMATION PROCESSING SYSTEM SIMULATOR

PAGE 68

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
DEPART QUEUE: FACILITY = CU2314:
SET STATUS: FACILITY = CU2314:
    STATUS = INUSE:
SETIF: FACILITY = CU2314:
C
QUEUE: FACILITY = SELFCT(1):
WAIT FACILITY: FACILITY = SELECT(1):
    STATUS = AVAIL:
DEPART QUEUE: FACILITY = SELECT(1):
C
C *** THE CHANNEL IS ACQUIRED MOMENTARILY TO PERFORM SEEK
C
    SEEK: ACCESS MECHANISM = AM,
        CYLINDER = CYL:
    WAIT INPUT OUTPUT: IOTYPE = SEEK:
C
C *** SFEK IS COMPLETE. RF-ACQUIRE THE CHANNEL AND TRANSFER THE DATA
C
CTRACEWRITE(4,131)
131 FORMAT(' ',**MSG CHRCM AFTER SEEK IS DONE *)
QUEUE: FACILITY = SELECT(1):
WAIT FACILITY: FACILITY = SELECT(1):
    STATUS = AVAIL:
DEPART QUEUE: FACILITY = SELECT(1):
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
1609		1543
1610		1544
1611		1545
1612		1546
1613		1547
1614		1548
1615		1549
1616		1550
1617		1551
1618		1552
1619		1553
1620		1554
1621		1555
1622		1556
1623		1557
1624		1558
1625		1559
1626		1560
1627		1561
1628		1562
1629		1563
1630		1564
1631		1565
1632		1566

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
INPUT CARD IMAGE
SET STATUS: FACILITY = SELECT(1),
STATUS = INUSE:
CHSTAT(1) = INUSE
SEIZE: FACILITY = SELECT(1):
C *** TEST THE TYPE OF I/O TRANSFER
C
IF (RDWT .NE. 1) GO TO 150
DATA TRANSFER: TYPE = READ,
LATENCY = YES,
VOLUME = VOL,
DATASET = DS.F,
PHYSICAL RECORD LENGTH = PRECL:
GO TO 175
170 CONTINUE
DATA TRANSFER: TYPE = WRITE,
LATENCY = YES,
VOLUME = VOL,
DATASET = DS.F,
PHYSICAL RECORD LENGTH = PRECL:
175 CONTINUE
WAIT INPUT OUTPUT: ICTYPE = DATA TRANSFER:
C *** REPEAT ALL OF THE HELD FACILITIES AND SOLIT
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT LEV	INPUT SEQ-NO	MEMBER REF-NO
1633			1567
1634			1568
1635			1569
1636			1570
1637			1571
1638			1572
1639			1573
1640			1574
1641			1575
1642			1576
1643			1577
1644			1578
1645			1579
1646			1580
1647			1581
1648			1582
1649			1583
1650			1584
1651			1585
1652			1586
1653			1587
1654			1588
1655			1589
1656			1590

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

PAGE 70

DATE ... 08/28/79

```

.....10.....20.....30.....40.....50.....60.....70.....80
C
    RELEASE: FACILITY = SELECT(1);
    SET STATUS: FACILITY = SELECT(1);
        STATUS = AVAIL;
    CHSTAT(1) = AVAIL
    RELEASE: FACILITY = CU2314;
    SET STATUS: FACILITY = CU2314;
        STATUS = AVAIL;
    RELEASE: FACILITY = AM;
    SET STATUS: FACILITY = AM;
        STATUS = AVAIL;
    GO TO 900
C
C *** NOW HANDLE TAPE I/O
C
    200 IF (DTYPE .NE. 2) GO TO 800
    QUEUE: FACILITY = AM;
    WAIT FACILITY: FACILITY = AM,
        STATUS = AVAIL;
    DEPART QUEUE: FACILITY = AM;
    SET STATUS: FACILITY = AM,
        STATUS = INUSE;
    SEIZE: FACILITY = AM;
C
.....10.....20.....30.....40.....50.....60.....70.....80

```

INPUT SEQ-NO	ALT LEV	INPUT SEQ-NO	MEMBER REF-NO
1657			1591
1658			1592
1659			1593
1660			1594
1661			1595
1662			1596
1663			1597
1664			1598
1665			1599
1666			1600
1667			1601
1668			1602
1669			1603
1670			1604
1671			1605
1672			1606
1673			1607
1674			1608
1675			1609
1676			1610
1677			1611
1678			1612
1679			1613
1680			1614

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

	INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
.....10.....			
.....20.....			
.....30.....			
.....40.....			
.....50.....			
.....60.....			
.....70.....			
.....80.....			
QUEUE: FACILITY = CU2804:	1681		1615
WAIT FACILITY: FACILITY = CU2804:	1682		1616
STATUS = AVAIL:	1683		1617
DEPART QUEUE: FACILITY = CU2804:	1684		1618
SET STATUS: FACILITY = CU2804:	1685		1619
STATUS = INUSE:	1686		1620
SEIZF: FACILITY = CU2804:	1687		1621
	1688		1622
	1689		1623
	1690		1624
	1691		1625
	1692		1626
	1693		1627
	1694		1628
	1695		1629
	1696		1630
	1697		1631
	1698		1632
	1699		1633
	1700		1634
	1701		1635
	1702		1636
	1703		1637
	1704		1638

.....1.....20.....30.....40.....50.....60.....70.....80

```

DATE ... 08/28/79
.....10.....20.....30.....40.....50.....60.....70.....80
C
C *** TEST TYPE, IF TRANSFER THEN DO IT
C
IF (PDNT .NE. 1) GO TO 350
DATA TRANSFER: TYPE = READ,
DATA SET = DS.F,
VOLUME = VOL,
PHYSICAL RECORD LENGTH = PRECL:
GO TO 400
350 CONTINUE
DATA TRANSFER: TYPE = WRITE,
DATA SET = DS.F,
VOLUME = VOL,
PHYSICAL RECORD LENGTH = PRECL:
GO TO 400
C
C *** WAIT FOR COMPLETION AND RELEASE FACILITIES AND SPLIT
C
400 CONTINUE
WAIT INPUT OUTPUT: IOTYPE = DATA TRANSFER:
RELEASE: FACILITY = SELECT(SEL$W):
SET STATUS: FACILITY = SELECT(SEL$W),
STATUS = AVAIL:
CMSTAT(SEL$W) = AVAIL
.....10.....20.....30.....40.....50.....60.....70.....80

```

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```

.....10.....20.....10.....INPUT CARD IMAGE.....60.....70.....80
RELEASE: FACILITY = CU2804;
SFT STATUS: FACILITY = CU2804;
      STATUS = AVAIL;
RELEASE: FACILITY = AM;
SFT STATUS: FACILITY = AM;
      STATUS = AVAIL;
GO TO 900
C
C
C ***OK BILL HERE IT IS AND I THINK IT WILL WORK***
C
      RDD IF(OVTYPE .NE. 4) GO TO 900
C UNSPEC DEVICE TYPE
C GET THE TRANSFER RATE FROM THE $CFDEV ARRAY
      IVTYP = $CFDEV(CFDEV+7)
      GO TO (R10,R12,R14,R16,R18),IVTYP
C ERROR - PUT ASSUME THE TYPE IS INTEGER
C
C INTEGER IS SPECIFIED - ASSIGNMENT WILL CONVERT IT TO REAL
      R10 TRATE = $CFDEV(CFDEV+9)
      GO TO R20
C REAL IS SPECIFIED
      R12 CALL $RLE (TRATE,$CFDEV(CFDEV+9))
      GO TO R20

```

```

.....10.....20.....30.....40.....50.....60.....70.....80

```

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

DATE ... 04/28/79

INPUT SEQ-NO	ALT LEV	INPUT SEQ-NO	MEMBER REF-NO
1753		1687	
1754		1688	
1755		1689	
1756		1690	
1757		1691	
1758		1692	
1759		1693	
1760		1694	
1761		1695	
1762		1696	
1763		1697	
1764		1698	
1765		1699	
1766		1700	
1767		1701	
1768		1702	
1769		1703	
1770		1704	
1771		1705	
1772		1706	
1773		1707	
1774		1708	
1775		1709	
1776		1710	

.....10.....20.....30.....40.....50.....60.....70.....80

C SYSPAR VALUE IS SPECIFIED

814 TRATE = SYSPAR(SCFDEF(CFDV+8))

GC TO 820

C PROCEDURE REFERENCE IS SPECIFIED

816 CALL SCFPR(SCFDEF(CFDV+8))

TRATE = SYSCOM(1)

GC TO 820

C GLOBAL PROCEDURE REFERENCE IS SPECIFIED

C EPROP - GPRCC IS NOT IMPLEMENTED

818 TRATE = 10

C TRATE IS ASSUMED TO BE SPECIFIED IN CHARACTERS PER SECOND

C COMPUTE THE TIME REQUIRED TO TRANSFER PRECL CHARACTERS (BYTES)

C IN MSEC TIME UNITS

820 TME = (PRECL/TRATE) * 1000

QUEUE: FACILITY = AM;

WAIT FACILITY: FACILITY = AM;

STATUS = AVAIL;

DEPART QUEUE: FACILITY = AM;

SET STATUS: FACILITY = AM;

STATUS = INUSE;

SFIZE: FACILITY = AM;

C HOLD THE AM FOR THE DURATION OF THE DATA TRANSFER

C NOTE - THE IPSS DATA TRANSFER STATEMENT ISN'T SUITABLE FOR

C UNIT RECORD OR UNSPEC TYPE DEVICES

.....10.....20.....30.....40.....50.....60.....70.....80

INFORMATION PROCESSING SYSTEM SIMULATOR
STANDARD INPUT STREAM LISTING

PAGE 75

DATE ... 08/28/79

.....10.....20.....30.....40.....50.....60.....70.....80

PROCESS: TIME = TIME;

RELEASE: FACILITY = AM;

SET STATUS: FACILITY = AM, STATUS = AVAIL;

C

C *** ALL DONE. SO RELEASE THIS SERVICE AND SPLIT

C

900 CONTINUE

RELEASE: FACILITY = CHPGM;

.....

END: ENDO SERVICE;

PROCEDURE: NAME = PRUF,

TYPE = SUBROUTINE,

PARAMETER LIST = (RETCU, PUTC);

C

C PROCEDURE FILE READS AND PROCESSES INTO TABLES

C THE INPUT RECORDS DEFINING SYSTEM AND USER FILES

C

INTERGER NAME, (LX5Z,CHT15),DIMSET,OTRPF,OTRPU(7),ETYPF,FILE

INSTR = CHT15,OTRPF,INSTRX,OTRPU,OTRPU,OTRPU,OTRPU

.....

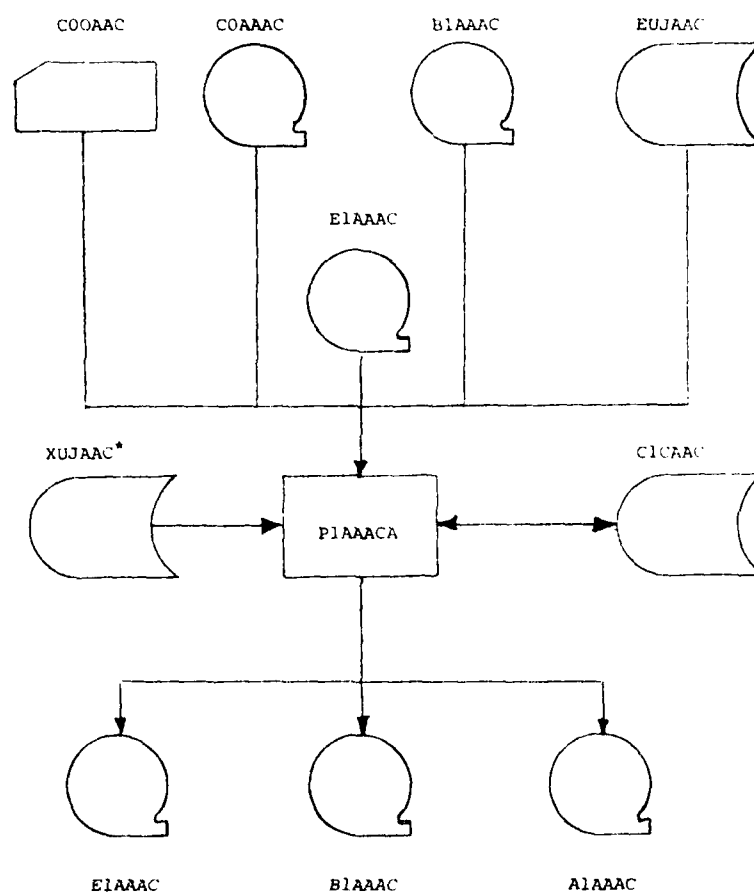
INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
1777		1711
1778		1712
1779		1713
1780		1714
1781		1715
1782		1716
1783		1717
1784		1718
1785		1719
1786		1720
1787		
1788		
1789		
1790		
1791		
1792		1721
1793		1722
1794		1723
1795		1724
1796		1725
1797		1726
1798		1727
1799		1728
1800		1729

APPENDIX C

SIDPERS JOBSTEPS P1A THROUGH P1G

PROCESSING CHARACTERISTICS

This appendix contains the basic system flow charts for SIDPERS programs P1A, P1B, P1C, and P1G (Figures C-1 through C-4 respectively). For each program, Table C-1 presents a brief file description, record length and blocking factor specifications, the type of storage media on which the file resides, and an indicator of file use (input to the program, output from the program, or both input and output).



* X = A,B,C,E,F,G,H

Figure C-1. PIA File Identification

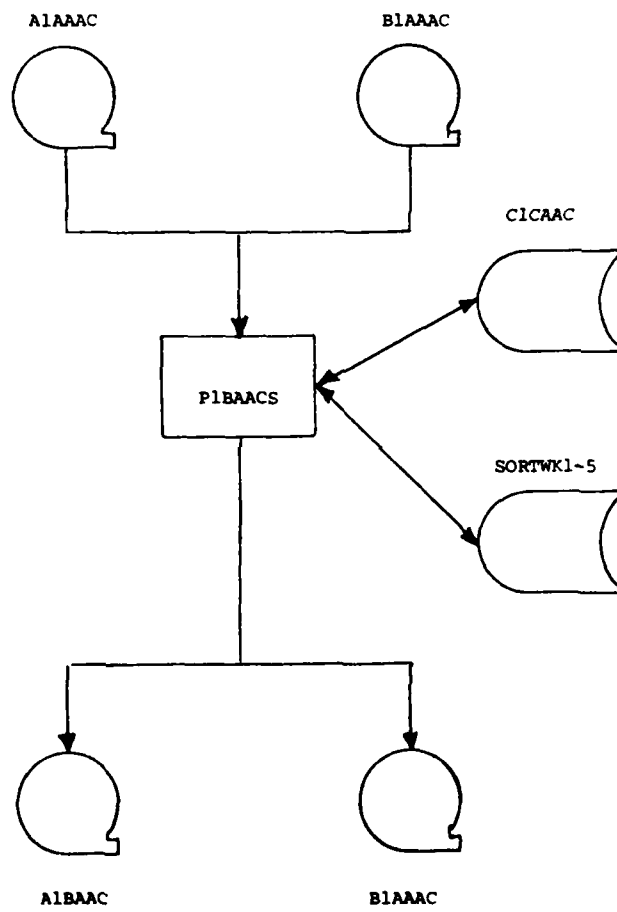


Figure C-2. PlB File Identification

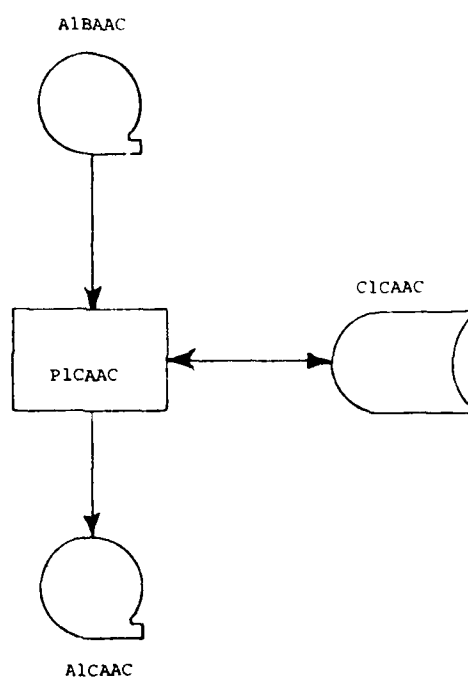
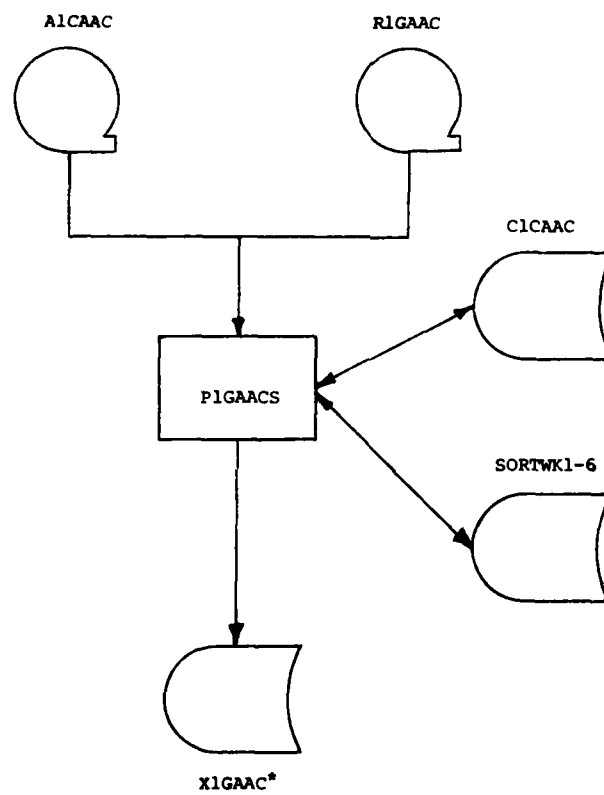


Figure C-3. PLC File Identification



* X = A,B,C,E,F,G,I,J,K,M,N,Q,R

Figure C-4. PLG File Identification

Table C-1. File Characteristics for Programs P1A Through P1G

File Name	Media	Input/ Output	Description	Logical Record Length	Blocking Factor
<u>P1A</u>					
COOAAC	Card/Tape	I	Optional Input	80	1
B1AAAC	Tape	I	Input Stacker File	100	40
COAAAC	Tape	I	Transaction	80	1
AlAAAC	Tape	O	Class-sched trans file	132	10
ElAAAC	Tape	O	SIDPERS trans history	80	10
BlAAAC	Tape	O	Output Stacker file	100	40
C1CAAC	Disk	I/O	Edit table file	506	2
XUJAAC (X=A,B,C, E,F,G,H)	Disk	I		80	1
<u>P1B</u>					
AlAAAC	Tape	I	Class-sched trans	132	10
B1AAAC	Tape	I	Monthly	100	40
C1CAAC	Disk	I/O	Edit table file	506	2
AlBAAC	Tape	O	Sorted CS trans	132	10
B1AAAC	Tape	O	SSF	100	40
SORTWK1-5	Disk	I/O	Sortwork File	132	12
<u>P1C</u>					
AlBAAC	Tape	I	Sorted CS trans	132	10
C1CAAC	Disk	I/O	Edit table file	506	2
AlCAAC	Tape	O	Edited trans	286	8

Table C-1 Continued.

File Name	Media	Input/ Output	Description	Logical Record Length	Blocking Factor
<u>PLG</u>					
AlCAAC	Tape	I	Edited trans	286	8
RlCAAC	Tape	I/O	Recycle trans	286	8
ClCAAC	Disk	I/O	Edit table	506	2
SORTWK1-6	Disk	I/O			
*lCAAC	Disk	0	A	280	6
where * = A,B,C,E, F,G,I,J,K,M,N,Q,R			B	285	8
			C	285	8
			E	285	8
			F	80	20
			G	125	25
			I	84	20
			J	90	25
			K	280	6
			M	80	20
			N	80	20
			Q	84	41
			R	286	8

APPENDIX D

EXAMPLES OF IAPS INPUT TABLES

This Appendix contains a complete listing of the Application File Table (Figure D-2) and System File Table (Figure D-3) for the first four jobsteps of SIDPERS. It also contains a partial listing of the Application Processing Table (Figure D-4), namely that portion which represents the first two jobsteps of SIDPERS (PIA and PIB). For the reader's convenience, in Figure D-1, we give an explanation of the headings for the Application and System File Tables.

Application File Table

FILE - a user given unique identifier for each application file

LRECL - Logical record length of application file

BLKSZ - Blocksize

#RECS - Number of logical records to be processed

System File Table

FILE - Same as above, to be used as a cross reference between the two file tables

VOLUME - Physical unit type (D for disk, T for tape, C for console) and unit number

LRECL - Logical record length from system's point of view

BLKSZ - Physical blocksize

#RECS - Number of records on file

%PE - Percentage of records on primary extent

#SE - Number of secondary extents

K/U - Placement known (K) or unknown (U)

TYPE - Primary extent (P), index extent (I), overflow (O), or VTOC (V), for disk files only

PLACEMENT - Actual placement of disk file, if known, given in low cylinder - low track address to high cylinder - high track address

Figure D-1. Explanation of Headings in Figures D-2 and D-3

FILE TABLE				COMMENTS	
FILE	LRCL	BLK SZ	#RECS	INPUT TO	PIAACA
1	80	80	554	OPTIONAL CARD INPUT	COAACC
2	80	80	661	INPUT TRANSACTIONS	COAACC
3	80	80	1	D P E N E D	AUJAAC
4	80	80	1		BUJAAC
5	80	80	1		CUJAAC
6	80	80	1		CUJAAC
7	80	80	1	R E A D	CUJAAC
8	80	80	1		CUJAAC
9	80	80	1		CUJAAC
10	506	506	987		CUJAAC
11	100	100	2111	INPUT & OUTPUT	CUJAAC
12	132	112	1249		CUJAAC
13	80	80	1171	OUTPUT PIAACA	CUJAAC
14	132	132	1249		CUJAAC
15	132	132	100	OUTPUT PIAACS	CUJAAC
16	132	132	100	S O R T	CUJAAC
17	132	132	100		CUJAAC
18	132	132	100	W O R K	CUJAAC
19	132	132	100		CUJAAC
20	286	286	1249	FILES	CUJAAC
21	286	286	50	OUTPUT PIAACA	CUJAAC
22	506	506	100	IN & OUTPUT FOR	CUJAAC
23	280	280	0	PIAACS	CUJAAC
24	285	285	30	OUTPUT PIAACS	CUJAAC
25	285	285	1210		CUJAAC
26	80	80	0		CUJAAC
27	125	125	0		CUJAAC
28	84	84	0		CUJAAC
29	90	90	0		CUJAAC
30	280	280	2		CUJAAC
31	80	80	0		CUJAAC
32	80	80	0		CUJAAC
33	84	84	0		CUJAAC
34	286	286	100	S O R T	CUJAAC
35	286	286	100		CUJAAC
36	286	286	100	W O R K	CUJAAC
37	286	286	100		CUJAAC
38	286	286	100	FILES	CUJAAC
39	286	286	100		CUJAAC
40	286	286	100	CONSOLE	CUJAAC
41	40	40	100		CUJAAC
42	100	100	2111		CUJAAC

Figure D-2. Application File Table for SIDDERS

Figure D-3. System File Table for SIDPERS

```

*
*      S I D P E R S
*
*      JOBSTEP - PIAAACA
*
*      INPUT ONLY
*      1 CARD/TAPE FILE
*      - CDDAAC
*
*      1 TAPE FILE
*      - CDDAAC
*
*      7 DISK FILES
*      - AUJAAC, BUJAAC, CUJAAC, EUJAAC, FUJAAC, GUJAAC, HUJAAC
*
*      INPUT & OUTPUT
*      1 TAPE FILE
*      - BIAAAC
*
*      1 DISK FILE
*      - CICAAC
*
*      OUTPUT ONLY
*      2 TAPE FILES
*      - AIAAAC, EIAAAC
*
*
*      EXEC      BEGIN PIAAACA
*
*      INPUT PROCESSING
*
*      I          CDDAAC - CARD/TAPE OPTIONAL CARD INPUT
*      01 X S 100.
*
*      I          CDDAAC - INPUT TRANSACTIONS
*      02 Y S 100.
*
*      *I          AUJAAC
*      * 03 A S 100.
*      *I          BUJAAC

```

Figure D-4. Application Processing Table for SIDPERS (Jobsteps PIA and PIB)

```

* 04 B S 100.
*I          CUJAAC
* 05 C S 100.
*I          EUJAAC
* 06 E S 100.
*I          FUJAAC
* 07 F S 100.
*I          GUJAAC
* 08 G S 100.
*I          HUJAAC
* 09 H S 100.
*
I          CICAAC - EDIT TABLE FILE
10 T I .44      (FOUR READS OF FILE ONLY - .44 TIMES 987 = 4)
*
I          BIAAAC - STACKER FILE
11 Z S 100.
*
*
P          CPU PROCESSING
X          COMPARE   SELECT
*
*
O          MESSAGE TO CONSOLE
41 M S 23.
*
*
*          OUTPUT PROCESSING
*
O          CICAAC - EDIT TABLE FILE - 2 WRITES
10 U I .22
*
*
O          AIAAAC - CLASS SCHED TRANS FILE
12 X S 44.4
*
O          EIAAAC - SIPPERS TRANSACTION HISTORY
13 X S 47.4
*
*
O          AIAAAC - CLASS SCHED TRANS FILE

```

Figure D-4 Continued.

```

12 Y S 51.0
*
D          E1AAAC - SIDPERS TRANSACTION HISTORY
13 Y S 56.5
*
*
C          B1AAAC - STACKER FILE
42 Z S 58.6
*
C          A1AAAC - CLASS SCHED TRANS FILE
12 Z S 2.49
*
D          E1AAAC - SIDPERS TRANSACTION HISTORY
13 Z S 2.65
*
D          OPERATOR DELAY (RESPONSE TO CONSOLE, OR TAPE MOUNT)
10000. 10000. 1.0
*
*
*          END JOBSTEP P1AAACA
*
EOP
*
*
*
*          JOBSTEP - P1EAAACS
*
*          INPUT ONLY
*          1 TAPE FILE
*          - A1AAAC
*
*          1 DISK FILE
*          - C1CAAC
*
*          1 TAPE FILE (MONTH END ONLY)
*          - B1AAAC
*
*          INPUT & OUTPUT
*          1 TAPE FILE
*          - B1AAAC
*
*          OUTPUT ONLY

```

Figure D-4 Continued.


```

EOP
*
I          A1AAAC - CLASS SCHED TRANS FILE
12 X S 29.7
*
P
X          COMPARE   SORT
*
D
17 Y S 370.
*
ECP
*
*          END INITIAL SORT PHASE - NOW MERGE
*
I
15 X S 440.
I
16 X S 440.
I
17 X S 370.
*
P
X          MERGE
*
D          A1BAAC - SORTED CLASS SCHED TRANS FILE
14 X S 35.3
D
14 X S 35.3
G
14 X S 29.7
*
C          OPERATOR DELAY (RESPONSE TO CONSOLE, OR TAPE MOUNT)
10000. 10000. 1.0
*
*
*
*
*          END JOBSTEP P1BAACS
*
EOP
*

```

Figure D-4 Continued.

APPENDIX E

THE MODEL LIBRARY

What follows is a complete list of the currently existing members of the model library, followed by a brief discussion of those members which the User would have to be familiar with in order to run models.

\$GADRU - IPSS Get Address routine, modified for IAPS methodology

BUFMR - Buffer manager

CHPGM30 - Channel program for IBM 360/30 hardware

CHPGM47 - Channel program for Honeywell Model 47 hardware

HDWM30 - Hardware specifications for IBM 360/30, all variants

HDWM47 - Hardware specifications for Honeywell Model 47, all variants

M30A2 - Hardware configuration A2 (See Table 5-3)

M30A3 - Hardware configuration A3

M30A4 - Hardware configuration A4

M47B2 - Hardware configuration B2

RDGP - Reads Application Processing Table and prepare it for processing

RSUF - Reads System and Application File Tables, and set up Index Tables

SERVICES - Application program processing

SIDIOPT - SIDPERS Application Processing Table (for the first four jobsteps of SIDPERS)

SIDSFT - System File Table for SIDPERS

SIDSFTB2 - System File Table for SIDPERS for configuration B2

SIDSFT2T - System File Table for SIDPERS with 2 tape files (other tape files transferred to disk)

SIDVFT - Application File Table for SIDPERS

SID30A1 - Hardware configuration A1

SID47B1 - Hardware configuration B1

STORAGE - Generalized database description

The following members form the core of the IAPS methodology and would be in every model; thus they need not concern the User.

\$GADRU
 BUFMR
 RDGP
 RSUF
 SERVICES
 STORAGE

The next group of members define the hardware configuration and thus require a User choice. The brackets indicate that a choice of one and only one must be made.

STD30A1
 M30A2
 M30A3
 M30A4
 STD47B1
 M47B2

If one of the first four configurations is chosen, then hardware specifications will come from HWDM30 and CHPGM30; otherwise, HDWM47 and CHPGM47 will be used.

The second and final decision made by the User before submitting a run involves the workload, or loading. At present, there is only one Application Processing Table, SIDIOPT, which models the first four jobsteps of SIDPERS, and there is only one Application File Table, SIDUFT, which specifies the file characteristics of SIDPERS files from the Cobol programmer's point of view. However, there are 3 choices for System File Table, namely SIDSFT, SIDSFTB2, and SIDSFT2T. The first choice, SIDSFT, places all files on disk and tape units exactly as current Army practice, while the latter two offer slight variations on file placement to accommodate different hardware configurations.

In summary, to run models the User would have to make two choices, one on the hardware configuration desired and the other on the desired loading.

APPENDIX F

GUIDE TO PREPARING IAPS INPUT

This appendix contains detailed formatting information for the three input tables required to use the IAPS methodology. The three tables are the Application File Table, the System File Table, and the Application Processing Table. Examples of these tables for SIDPERS are in Appendix D.

Application File Table Format

The application file table is a description of each file from the application programmer's point of view.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1-2	Any number from 01 to 50, no two identical	A unique file identifier
3	Blank	
4-7	Logical record length (in bytes)	
8	Blank	
9-14	Blocksize (in bytes)	
15	Blank	
16-21	Number of logical records in file	
22-72	Modeler comments	

System File Table Format

The system file table is a description of each file from the hardware point of view.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1-2	Any number from 01 to 50	The file identifier from the application file table.
	0	Any system file not directly referenced by a user's program
3	Blank	
4	T, D, or C	T - Tape D - Disk C - console
5	Blank	
6	Device unit number (01 to 50)	
7	Blank	
8-11	Logical record length	
12	Blank	
13-18	Blocksize	
19	Blank	
20-25	Number of logical records in file	
26	Blank	
27-29	Percent of records in primary extent	
30	Blank	

System File Table Format (continued)

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
31-32	Number of secondary extents	
33	Blank	
34	K or U	K - known placement U - unknown placement
35	Blank	
36	I, P, O, V or Blank	I - index extent P - prime extent O - overflow extent V - VTOC Blank - prime extent (for tape files)
37	Blank	
38-40	Low cylinder address (LCA)	The pair LCA-LTA gives the beginning address of the file on disk
41	Blank	
42	Low track address (LTA)	
43	Blank	
44-46	High cylinder address (HCA)	The pair HCA-HTA gives the ending address of the extent allocated to the file
47	Blank	
48-49	High track address (HTA)	
50-72	Modeler comments	

Application Processing Table Format

The Application Processing Table describes the processing performed by application systems. This table allows the specification of I/O activities, CPU processing and delays. Table entries are easily grouped into identifiable packets of real world activities (job steps and jobs), and allow for user comments.

- a) Comment cards may appear anywhere within the application processing input except between an I, O, or D processing specification card and its associated definition card.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1	*	Card is ignored by processor
2-72	User comments	

- b) Delimiter cards mark the beginning and end of processing and execution groups.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1	Blank	
2-6	EXEC	Begin an execution group
2-5	EOP	End processing group
7-72	User comments	

- c) Processing specification cards model the input, output, and processing activities within each processing group.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1	Blank	
2	I	Input file Zero or more occurrences Must precede all "P" and "O" cards within a processing group
	P	Processing option One occurrence Must precede all "O" cards within a processing group
	O	Output file Zero or more occurrences Must follow "P" card
	D	Delay option No more than two occurrences Must be the first and/or the last processing specification card in a processing group

3-72 User comments

- d) An I/O definition card must follow each I or O specification card.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1-2	Blank	
3-4	Any integer from 01 to 99	Application file number
5	Blank	

d) continued.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
6	Blank	Nonconcurrent activity
	Any non-blank alphanumeric character	Concurrent activity
7	Blank	
8	S	Sequential access
	R	Random access
	I	ISAM file
	V	VSAM file
9	Blank	
10-12	Any integer from 0 to 100	Percent of records processed
13-72	User Comments	

e) At least one P definition card must follow the P specification card.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1-2	Blank	
3	Blank	Processing not concurrent with any I/O
	Non-blank	Processing concurrent with associated I/O
4-9	Blank	
10-19	Any of the codes:	Defines processing activity
	SORT	(Each code must start in col 10)
	MERGE	
	COMPUTE	
	EDIT	
	UPDATE	
	SELECT	
	REPORT	
	USEREXIT	

e) (continued)

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
20-29		Coded as col 10-19
30-39		
40-49		First blank field terminates card
50-59		
60-69		

f) Exactly one D definition card must follow each D specification card.

<u>Column</u>	<u>Code</u>	<u>Explanation</u>
1-2	Blank	
3-9	Any nonnegative decimal number	Estimated minimum delay (All 3 numbers should have a decimal point)
10	Blank	
11-17	Any nonnegative decimal number	Estimated maximum delay
18	Blank	
19-25	Any number between 0.0 and 1.0	Probability of a positive delay

APPENDIX G

GRASP ACCOUNTING DATA

The following is a partial listing of the measurement data provided by the GRASP accounting package. This list is included in this report to indicate the wide variety and usefulness of GRASP step accounting data to computer simulation projects.

Table G-1. GRASP Accounting Data

Partition	- identifies the partition in which the job executed
Time-on	- time of day the job began
Time-off	- time at which the job ended
Duration	- time the job occupied the partition
Non-MPS duration	- time the job would have run without interference
Interference duration	- time this job was interfered with by multiprogramming activity
CPU time	- time spent by this job executing CPU instructions
Operator duration	- time spent by this job in wait states of 3 seconds or longer
I/O wait time	- time spent waiting for data transfer to complete
Phase loads	- total fetches or loads performed by this job
Time waiting for LTA	- total time waiting for access to the transient area
Time using LTA	- total time the LTA was used by this job
Lines spooled	- number of print lines produced by this job
Cards spooled in	- number of input cards spooled for this job
Cards spooled out	- number of cards punched and spooled for this job
Start I/O counts	- the number of input or output requests issued for each symbolic logical unit used by this job

Table G-1 Continued.

I/O device usage time	- the total "device busy" time accrued on each I/O device used by this job
SYSRES usage time	- time spent by the job reading or writing on the SYSRES device
CPU utilization	- total time the CPU was active for any partition/purpose during the execution of this job
Channel activity	- total time each channel was "active"
CPU channel overlap	- overlap between CPU and channel activity
Core used	- total size of the program as loaded into storage