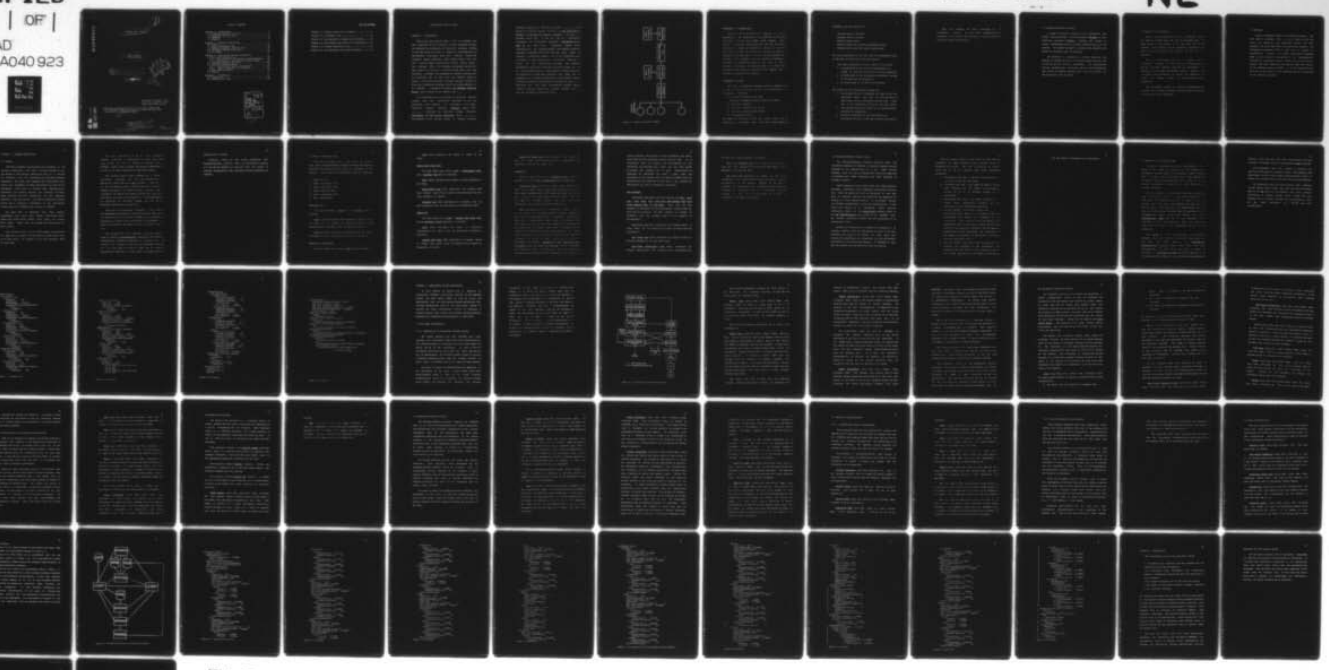
AD-A040 923 NORTH CAROLINA UNIV AT CHAPEL HILL DEPT OF COMPUTER S--ETC F/G 9/2
SATMODEL USER'S GUIDE,(U)
SEP 76 J D FOLEY, V L WALLACE

UNCLASSIFIED F30602-76-C-0117 NL

| OF |
AD
A040 923



END

DATE
FILMED
7-77

SATMODEL User's Guide*

James D. Foley
Victor L. Wallace

September 1976

71 p.

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina

* This work was sponsored by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441, under Contract No. F30602-76-C-0117.

409668

TABLE OF CONTENTS

LIST OF FIGURES

THE SATMODEL USER'S GUIDE

CHAPTER 1: INTRODUCTION

During the past several years a set of programs has been developed at the University of North Carolina to model and optimize the performance of satellite graphics systems of the form shown in Figure 1. Details of the modelling and optimization techniques have been reported previously [Graphics System Modeling, First Annual Report, UNC, June 1974. Graphics Model Verification, Second Annual Report, UNC, November 1975]. The programs, originally executed as batch jobs on an IBM 360 or 370 with OS/360, have now been converted, extended, and augmented to execute interactively on a Honeywell 645 or 6045 with the MULTICS operating system. This guide describes the basic concepts involved in using the interactive programs, which are herein referred to as SATMODEL. A companion document, The SATMODEL Reference Manual, gives details of the system's use.

The considerations in the design of satellite graphics systems have been extensively discussed in the two previously cited reports, and elsewhere [J.D. Foley, "Satellite Graphics Systems", Computer, August 1976. _____, "Software for Satellite Graphics Systems", Proceedings ACM 1973 Annual Conference, 76-80. _____, "An Approach to the Optimum Design of Computer Graphics

Systems", CACM 14 (6), 380-390, June 1971. _____, "The Design of Satellite Graphics Systems", in Data Structures in Pattern Recognition and Computer Graphics, A. Klinger, ed., Academic Press, 1976. A. van Dan, et. al., "Intelligent satellites for interactive graphics", Proceedings of the IEEE 62 (4), April 1974. "Operating system design considerations for microprogrammed minicomputer satellite systems"", Proceeding 1973 NEC.] The two basic issues are choosing capabilities (speed, size) of the subsystems shown in Figure 1.1, and selecting the appropriate "division of labor" for the applications program. The subsystem capabilities should be in balance one with another, so that no one subsystem acts as a bottleneck to system operations. The divisions of labor must recognize that there are two computers and storage hierarchies in the satellite system, and that programs and files must be assigned to them in an effective way. The basic optimization process aims to design a satellite system with minimum response time and cost not exceeding a specified bound.

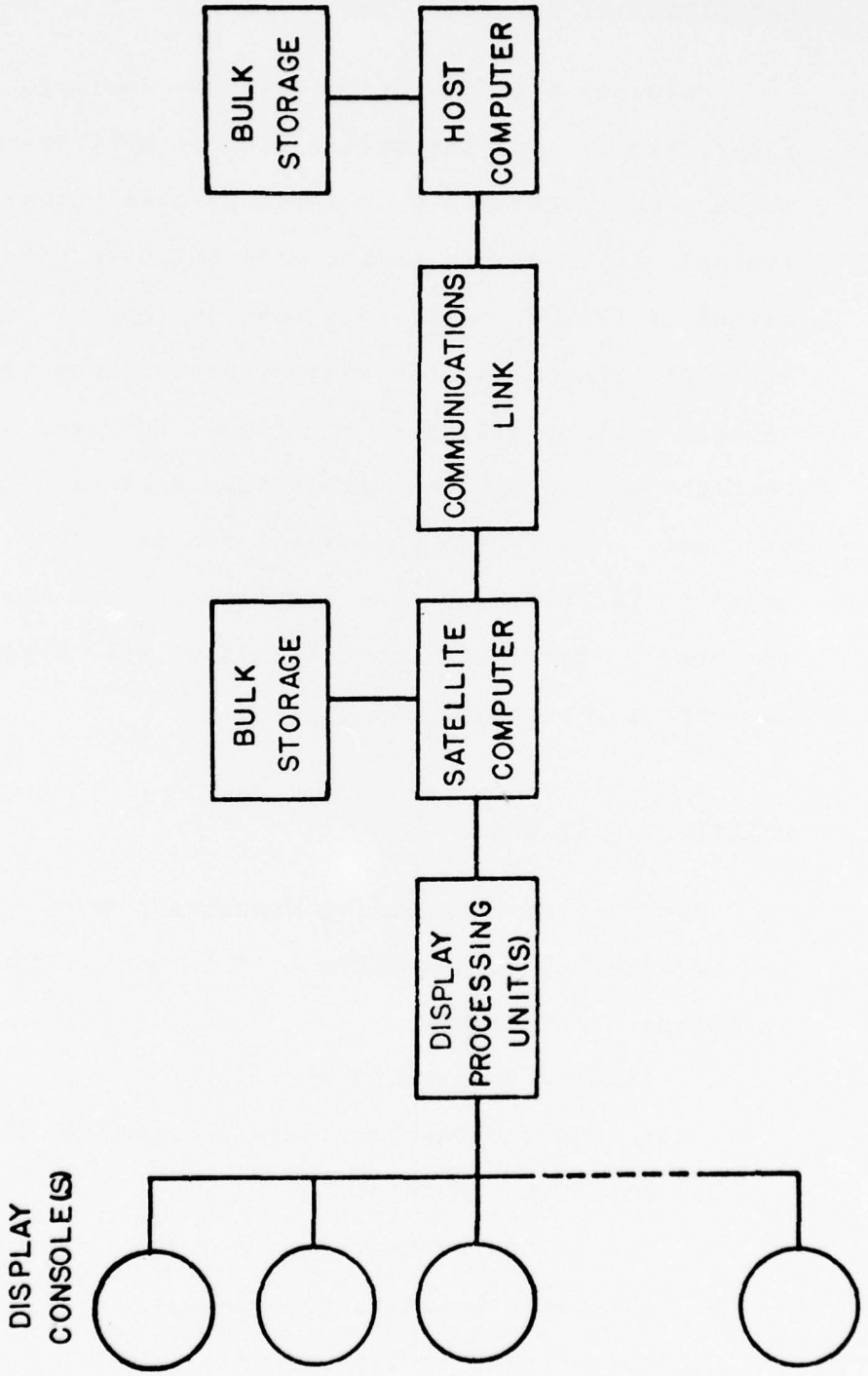


Figure 1.1 Graphics System Block Diagram

Definition of Response Time

Response time is measured for a typical, or average, interaction. An interaction is the activity of the system which occurs when the user invokes some command. Thus a typical interaction starts when the user hits the carriage return on the keyboard, lightpens an option on the menu, etc. At this point, the system must process the command, so program control will pass to some of the many subroutines or sections of the program, eventually returning information to the user. The system then waits for the user to initiate another interaction. The response time is the time between the user's last action in initiating the command and the response appearing on the display.

Definitions of Cost

The cost of a satellite graphics system is defined here to be the sum of the costs of five of the subsystems shown in Figure 1. They are:

1. Display Processing Unit (DPU)
   (including hardware-software tradeoff options),
2. Satellite Computer,
3. Satellite Computer Main Memory,
4. Satellite Computer Bulk Memory,
5. Communications Link.

The subsystem costs (and hence the system cost) can be specified in arbitrary units, and with various meanings

including, but not limited to:

purchase price in Dollars,
purchase price in Pounds,
monthly rental price,
monthly rental plus monthly maintenance price,
purchase price plus five year's maintenance.

The only requirement is that all costs be consistently given in the same units and with the same meaning.

The basic optimization process takes as its inputs:

1. A description of the cost and performance of a number of choices for each of the five subsystems.
2. A description of the interactive application program to be used with the satellite.
3. An upper bound on the cost of the system.

The results of the optimization process are:

1. The optimum choice of equipment for each of the five subsystems which, when used with the division of labor from 2 below, minimizes response time. Total cost of the subsystem is less than the upper bound.
2. The optimum division of labor for the above optimum selection of subsystems.
3. Detailed information on the performance and utilization of each of the five selected subsystems.

There are, however, yet other dimensions to the optimization process. We call these hardware-software tradeoffs, and selective specifications. They are described in the following two sections.

1.1 Hardware-Software Tradeoffs

A number of graphics functions can be implemented with either special-purpose display-processor hardware, or with satellite processor software. Examples are dynamic or static transformations and clipping, subpicturing, and text display. The special hardware is usually more expensive and faster in operation than the software.

The tradeoff in designing a minimum response time system is between the cost of the DPU hardware and the cost of one or more of the other subsystems. The optimization process automatically determines which of the alternatives will yield the fastest response time. This can be done for one or several such tradeoffs.

1.2 Selective Specification

Choosing the subsystems for use in a satellite system can involve issues other than cost and response time. The prime example of such issues is in the area of display quality, where criteria such as resolution, linearity, spot size, intensity levels, and flicker can all be very important.

We have accordingly provided a general culling (selection) mechanism to ensure that only those subsystems meeting specified minimum criteria are considered for selection by the basic optimization process. There is also a special-purpose mechanism to handle the important but unique and complex matter of flicker-free display capability.

The information needed to analyze hardware-software tradeoff and to effect selective specifications is included in the inputs to the optimizer.

1.3 SATMODEL

Using the SATMODEL system is a multistep process. The File Editor subsystem (FED) is first used to build two files, one describing the subsystem choices, and best hardware, the other describing the application program. The optimization subsystem (OPT) is then used to perform an optimization. Finally, the report generating subsystem (REP) is used to select that portion of the optimization's potentially voluminous output which is to be printed as a report. The three subsystems all operate under the control of the SATMODEL executive (EXEC). It is the detailed use of EXEC and its FED, OPT, and REP subsystems which is discussed in the companion document.

1.4 Remarks

In the following sections, we will often refer to names: program module names, file names, subsystem names, etc. These names, when entered into the system, may be up to 80 characters in length. However, when printed out to describe the results of the optimization process, the names may be truncated to 40 characters.

All measures of size, be it of main or backing store, programs, files, or messages, are given in units of bytes, while measures of time will be in units of seconds. Rates will always be specified in bytes per second, or seconds per byte.

Various abbreviations used are those of the companion The SATMODEL Reference Manual, referred to hereafter as SRM.

CHAPTER 2: HARDWARE DESCRIPTION

2.1 General

The basic hardware specifications are provided in the hardware descriptons (HF) file, of [SRM, Section 5.2.2]. The purpose of the hardware description file is to provide the optimizer with data concerning all available hardware components. This data, when combined with application data, yields the parameters by which performance and cost can be determined. This data is gleaned from manufacturers' catalogues and similar sources, and supplies the basic speed, capacity, and cost information for each component available from some vendor. This basic information changes slowly, and is generally independent of the application program which the SATMODEL user intends to optimize.

The basic data is separated into five groups, representing data links (DL), remote fast stores (RFS), remote bulk stores (RBS), remote CPU's (CPU), and remote DPU's (DPU). There must be at least one item in each of these groups.

Each hardware item in one of these groups is identified by a name entry, uniquely distinguishing it from other items in the same group. For example, no two data links may have the same name.

The name, consisting of 80 or fewer characters, normally identifies a manufacturer or source, and a model name, so that the item is readily recognized when the SATMODEL output lists, by name, the optimal components, and so that it is easily recalled for data modification.

Each hardware item is also supplied with a number of attributes, measuring its cost and performance. In particular, every item has a cost attribute. Units of cost can be any measure, such as dollars per month (rental), or dollars (purchase). The choice of unit, however, must be consistent throughout the hardware description file and any application file that is used with it. Performance measures are different for different groups, and will be more explicitly described in section 2.2.

One group in the hardware description file describes the performance attributes of the host computer (HC), but is not provided with a cost since SATMODEL does not provide for optimizing the choice among alternative host machines. This group of performance attributes is also described in section 2.2.

The hardware file also provides data for selection based on specific characteristics, supporting the "selective specification" capability described in section 1.2. The function and content of this data are described in section 2.4. If a user does not intend to make use of selective specification features, the data fields associated with this

feature may be ignored.

Similarly, there are data fields associated with hardware-software tradeoff which are described in section 2.3, and may be ignored by a user who does not intend to consider optimizations where hardware-software tradeoffs are intended.

2.2 Basic Performance Data

The basic performance data, that which is necessary even when hardware-software tradeoff features and selective specification features are not used, is described in this section. Each group has distinctive fields for this data.

1. Data Link (DL)
2. Remote Fast Store (RFC)
3. Remote Bulk Store (RBC)
4. Remote Computer (CPU)
5. Display Processor (DPU)
6. Host Computer (HC)

Data Link (DL)

For each data link, a speed and an overhead must be provided.

Speed (DLSP) represents the transmission rate of which the data link is capable in bytes per second. A speed specification is mandatory for each data link entry.

Overhead (DLO) represents the overhead time in seconds associated with one transmission across the data link.

Remote Fast Store (RFS)

For each remote fast store a size must be provided.

Size (RFSI) represents the number of bytes in the store.

Remote Bulk Store (RBS)

For each remote bulk store a size, a seek-latency time, and a transfer time must be provided.

Size (RBSI) represents the number of bytes available in the store.

Seek-Latency time (SLT) represents the average seek plus latency time ("access time") associated with the bulk store facility, in seconds.

Transfer time (TRT) represents the transfer time per byte associated with the bulk store, in seconds per byte.

Remote CPU

For each remote CPU a speed, a maximum fast store size, and an operating system size must be provided.

Speed (CSP) represents the number of high-level instructions per second that the processor is capable of executing.

Maximum fast store (MFS) represents the largest amount of remote fast store which the remote CPU is capable of supporting, in bytes.

Operating system size (OSS) represents the number of bytes of remote fast store which must be reserved for the operating system of the remote CPU.

Remote DPU

For each remote DPU a set of compatible CPU's must be listed, and a set of DPU instructions must be described.

Compatible CPU's (CCP) is a group whose entries are names which match entry names from the remote CPU group. These names identify those CPU's which can serve the given DPU. Reasons for omission of a CPU from such a list are: (1) inadequate software support for that particular combination, (2) incompatible word lengths, etc.

DPU instructons (DPI) is a group whose primary purpose is to serve the hardware-software tradeoff capability (sec. 2.3) and the flicker test in selective specificaion (sec. 2.4). Nevertheless, this group also provides data which allows the size of the DPU programs to be reflected in the allocation of remote fast store and allows the CPU support of the display processor to be reflected in remote store allocation and response time calculations. Each DPU instruction is listed by name, specifying a series of performance attributes: incremental cost, procedure size, instruction size, CPU time, and DPU time. These attributes are described fully in section 2.3. When not involved in hardware-software tradeoff, DPU instructions are given in a

single "option", and assigned a zero incremental cost (OIC), since they are not separately priced from the DPU. If an instruction does not require CPU intervention for its execution, the procedure size (OPS) and CPU time (OCPT) attributes are likewise set to zero. Instructions can sometimes be aggregated and given a class name and attributes if the average value of these attributes does not significantly vary from DPU to DPU, and if the application description file data is similarly aggregated.

Host Computer

Attributes describing the host computer are fast store size, bulk store size, bulk store seek-latency time, bulk store transfer time, and CPU speed. Since only one host computer is considered in the optimization, neither name nor cost data are supplied. The host computer is assumed to influence only the response time as its hardware is preconfigured.

Fast store size (FSS) represents the maximum number of bytes which can be occupied by programs and data from the application.

Bulk store size (BSS) represents the number of bytes of storage available in the host bulk store.

Bulk store seek-latency time (BSSL) represents the average seek/latency time ("access time") associated with

the host bulk store facility, in seconds.

Bulk store transfer time (BSTT) represents the transfer time per byte associated with the host bulk store. It is measured in seconds per byte.

CPU speed (HSP) represents the number of high level instructions per second that the host processor is capable of executing, on the average. Because it is used in response time calculations, it is important that this speed represent the capability seen by a satellite, suitably derated, if the CPU is time-shared to account for competition from other job streams.

2.3 Hardware-Software Tradeoff Data

The hardware-software tradeoff features allow the optimizing programs of SATMODEL to consider hardware options provided by the manufacturer of a DPU. These options normally take the form of instructions which are supported in hardware that might otherwise have been provided by software support.

Usual examples of such options deal with transformation hardware, subroutine jumps, character generations, scaling, etc. Since not every user will be interested in the same sets of possible options, and since they can generally be treated in a quite standard manner, no particular options are provided automatically in the file organization. A user lists the options to be considered, using suitably descriptive names, in the Instructions subgroup (DPUI) of the DPU Characteristics group (DPUC) of the hardware file. The entries (instruction names) used are identifiers of 80 characters or less.

Whether the instruction is supported by hardware or by software affects both the performance and cost of the DPU. Therefore each entry in the Remote DPU (DPU) group must contain data describing the incremental cost and performance attributes of the alternative options. In SATMODEL no more than two options are permitted for each instruction.

The DPU entries contain a group (DPUI) of instructions, identified by names matching those provided in the DPU Characteristics group. For each instruction so listed, there are one or two options, each having attributes specifying:

1. Incremental cost (OIC) - the cost of the option as an add-on to the DPU cost (DPC).
2. Procedure size (OPS) - the number of bytes of remote store required for this option to be available, whether or not it is actually invoked by the application.
3. Instruction size (OIS) - the number of bytes of remote store additionally required for each invocation of the instruction capability.
4. CPU time (OCPT) - the number of basic CPU instruction-times which the execution of this DPU instruction will require per parameter in the invocation. (The aggregate CPU time taken by this instruction will be this value, divided by CPU speed (CSP) for the given CPU, multiplied by the number of invocations of this instruction and multiplied by the number of parameters per invocation - both taken from the application file.)
5. DPU time (ODPT) - the time, inseconds, which the DPU requires per parameter in the invocation. (The aggregate CPU time taken by this instruction will be this value multiplied by the number of invocations

and the number of parameters per invocation.)

2.4 Selective Specification Data

In order to accomplish selective specification, there needs to be a way to specify, in the hardware file, performance measures which do not directly affect response time. Measures such as resolution for a DPU, availability of floating point or ruggedization for a CPU, or access methods (sequential, direct) for a bulk store, must be infered somewhere.

No such measures are explicitly built into the hardware file. Instead, a series of five groups (DLC, RFC, RBC, CPUC, DPUC) are provided in the file so that a user can provide names for each specification over which he might wish to select. Both names and the dimensional units by which the quantities are measured must be specified in these supplementary groups. These name entries are called characteristic names and can be any identifier of 80 characters or less. The dimensional units are attributes (DIM) and can also be any identifier of 80 or less characters.

The values of these specifications for individual hardware items are entered via the basic data groups (DL, RFS, RBS, CPU, DPU), through the supplementary characteristics subgroup (DLSC, RFSC, RBSC, CSC, DPSC, respectively). In each case, the value is given by providing a characteristic name entry which matches one of the characteristic names in the corresponding characteristic

subgroup (DLC, RFC, RBC, CPUC, DPUC, respectively) and then providing an integer value in corresponding *value* attribute.

The application data will provide a criterion for selecting from among the hardware items. Each hardware item whose value in one or more listed characteristics fails to exceed the criterion will be excluded from the optimization.

The application data will also provide data regarding typical display scenes which, when used with DPU instruction data in the *DPU* group of this hardware data file, permits calculation of display flicker rates. These are also compared with a criterion provided in the application file, and may cause elimination of a DPU-CPU pair from consideration.

2.5 An Example

Figure 2.1 is a listing of a hardware file which illustrates all the features we have described. The listing is indented with the same format that is used in SRM, section 5.2.

```
hardware file(hf)
   data links(dl)
      1200 BPS: 1 mile
         cost(dlc)=        90
         speed(dlsp)=      144
         overhead(dlo)=    0.010435
         supplementary characteristics(dlsc)
            Distance
               value(val)=        1
      4800 BPS: 1 mile
         cost(dlc)=       275
         speed(dlsp)=      575
         overhead(dlo)=    0.002609
         supplementary characteristics(dlsc)
            Distance
              value(val)=        1
      2,000,000 BPS: 1 mile
         cost(dlc)=       405
         speed(dlsp)=   250000
         overhead(dlo)=    0.000006
         supplementary characteristics(dlsc)
            Distance
               value(val)=        1
      19200 BPS: 11 miles
         cost(dlc)=       950
         speed(dlsp)=     2250
         overhead(dlo)=    0.000007
         supplementary characteristics(dlsc)
            Distance
               value(val)=       11
   remote fast stores(rfs)
      32K Bytes Extra Core
         cost(rfcs)=      130
         size(rfsi)=    32768
         supplementary characteristics(rfsc)
            Cycle Time
               value(val)=     1020
      64K Bytes Extra Core
         cost(rfcs)=      340
         size(rfsi)=    65536
         supplementary characteristics(rfsc)
            Cycle Time
               value(val)=     1020
```

Figure 2.1 A Hardware File

```
   112K Bytes Extra Core
      cost(rfcs)=       470
      size(rfsi)=    114688
      supplementary characteristics(rfsc)
         Cycle Time
            value(val)=      1020
remote bulk stores(rbs)
   RK11;1 RK05
      cost(rbcs)=       275
      size(rbsi)=   2457600
      seek-latency time(slt)=     0.070000
      transfer time(trt)=     0.000005
      supplementary characteristics(rbsc)
         Size
            value(val)= -2457600
remote cpus(cpu)
   DEC PDP 11/45 with Memory Management
      cost(ccs)=       750
      speed(csp)=     10000
      maximum fast store(mfs)=    131072
      operating system size(oss)=     10384
      supplementary characteristics(csc)
         Access Time
            value(val)=      3300
   DEC PDP 11/70
     cost(ccs)=      1050
      speed(csp)=     20000
      maximum fast store(mfs)=    131072
      operating system size(oss)=     10384
      supplementary characteristics(csc)
         Access Time
            value(val)=      3300
remote dpus(dpu)
   VG Series 3 20
      cost(dpc)=       578
      compatible cpus(ccp)
         DEC PDP 11/40 with Memory Management
         DEC PDP 11/45
         DEC PDP 11/40
         DEC PDP 11/45 with Memory Management
         DEC PDP 11/70
```

Figure 2.1 (continued)

```
instructions(dpi)
   Basic Operation Set
      Regular Display
         incremental cost(oic)=          0
         procedure size(ops)=          0
         instruction size(ois)=          3
         cpu time(ocpt)=          0
         dpu time(odpt)=      0.000014
      High Speed Display
         incremental cost(oic)=        114
         procedure size(ops)=          0
         instruction size(ois)=          3
         cpu time(ocpt)=          0
         dpu time(odpt)=      0.000007
   Subroutine Jump
      DPU stack
         incremental cost(oic)=          0
         procedure size(ops)=          0
         instruction size(ois)=          0
         cpu time(ocpt)=          1
         dpu time(odpt)=      0.000009
   Rotation
      Rotation Hardware Option
         incremental cost(oic)=        251
         procedure size(ops)=          0
         instruction size(ois)=          2
         cpu time(ocpt)=          0
         dpu time(odpt)=      0.000010
      Rotation Software Package
         incremental cost(oic)=         80
         procedure size(ops)=        512
         instruction size(ois)=          3
         cpu time(ocpt)=          2
         dpu time(odpt)=      0.000000
supplementary characteristics(dpsc)
   resolution
      value(val)=       1774
   colors
      value(val)=          1
   dimensionality of display
      value(val)=          2
dpu instructions(dpui)
   Subroutine Jump
   Basic Operation Set
   Rotation
```

Figure 2.1 (continued)

```
host computer(hc)
   installation name(hnm)=UNCCC IBM 360/75
   fast store size(fss)=   307200
   bulk store size(bss)= 30720000
   bulk store seek-latency time(bssl)=    0.038400
   bulk store transfer time(bstt)=    0.000001
   cpu speed(hsp)=    10000
data link characteristics(dlc)
   Distance
      dimensions(dim)=Miles
remote fast store characteristics(rfc)
   Cycle Time
      dimensions(dim)=Memory Cycles per Millisecond
remote bulk store characteristics(rbc)
   Size
      dimensions(dim)=Negative of size field (rbsi)
cpu characteristics(cpuc)
   Access Time
      dimensions(dim)=Number of memory accesses per millisecond
dpu characteristic descriptions(dpud)
   dpu characteristics(dpuc)
      resolution
         dimensions(dim)=Lines per Inch
      colors
         dimensions(dim)=Number of Colors (1=black and white)
      dimensionality of display
         dimensions(dim)=Number of Coordinates needed
                                 to specify a Point
```

Figure 2.1 (continued)

CHAPTER 3: APPLICATION PROGRAM DESCRIPTION

In this section we explain how to describe an interactive graphics application program to the modeling system. The file editor (FED) is used to insert the description into the application program description file. We first describe the parts of the description needed to perform the basic optimization process of designing a minimum response time system, then discuss hardware-software tradeoffs and selective specifications, in that order.

3.1 The Basic Optimization

3.1.1 Description of Application Program Modules

The system designer who uses SATMODEL must first organize the application under study into program modules. This modularization is essentially the first step toward the actual design of the application. As such, it must be recognized that there are many ways any given application can be modularized. The designer should attempt to develop a realistic modularization, since the optimum division of labor found by SATMODEL will be in terms of these modules.

The level of detail of modularization is completely at the discretion of the user. A very coarse, first level modularization might be done, or the final detailed modularization might be used instead. The graphics package which builds and modifies DPU programs and handles

interaction devices might be treated as a single large module, or it might be further broken down into its consitituent parts. The latter option would be necessary if the designer were considering the possibility of placing some graphics package functions on the host, and others on the satellite. However, too fine a modularization can result in very long execution times for the optimization phase. The most modules we have dealt with is about 40. There must be at least one module in the program description. Figure 3.1 shows a coarse modularization which is typical of many interactive graphics applications. The figure is the standard block diagram of a graphics program modularized to separate lexical, syntactic, and semantic processing.

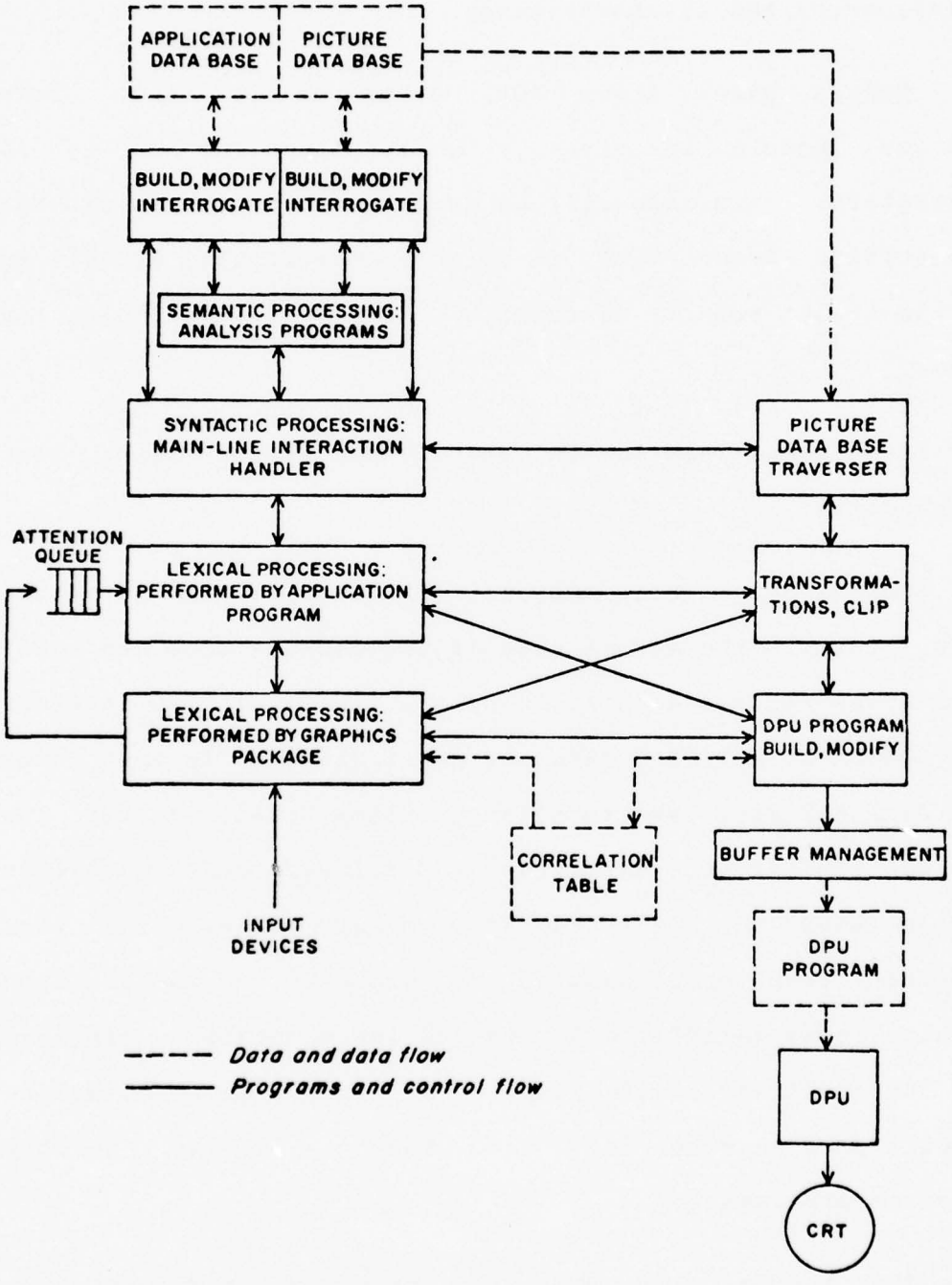


Figure 3.1 An Application Program Block Diagram

The detailed information required for each module is now described. For reference purposes, we give the FED group, entry and attribute names.

Module Name: group MOD, entry Module Name. Each program module is given a unique name of up to 80 characters. This name will be used in other parts of the application description to refer to the module, as well as in the output report describing the optimal division of labor.

Each module has several attributes, all of which must be specified.

Module Size: group MOD, entry Module Name, attribute MSIZ. This is the actual size of the machine code and local data areas for the module, in bytes. Areas of I/O buffers and COMMON or EXTERNAL data are not included. In most cases the designer will have to estimate these sizes, unless the application already exists and its relevant characteristics can be measured. In practice, module sizes will vary somewhat from one computer to another, dependent upon architectural details. The planned implementation language for the application must also be considered, since compilers usually produce more code than would a clever assembly language programmer.

The module size will be used when the optimizer considers different divisions of labor. For instance, some

modules, or combinations thereof, may require more main memory space than is available with the satellite computer.

Module Instructions: group MOD, entry Module Name, attribute MINS. This is the average number of instructions executed each time the module is invoked (called). The meaning of the word "instructions" is completely at the designer's discretion, and might equally well be called "units of work". These units can be thought of as low-level primitives such as add, subtract, text, move, etc., or as higher-level primitives like list insertion and searching, sorting, or adding up a collection of numbers.

The instruction counts are used by SATMODEL to calculate the average execution time of the module (exclusive of all I/O) on both the host and satellite. In the former case, a remote computer's speed, in instructions per second (or work units per second) is used; in the latter case, the host computer's speed. These speeds are taken from the hardware file. It is thus the designer's responsibility to estimate (or measure) the instruction counts and speeds so that correct execution times are arrived at when the count is divided by the speed.

Memory Management: group MOD, entry Module Name, attribute MMEM. The designer can specify one of three different memory management algorithms to be used with each module in the model or he can have SATMODEL select the best algorithm. The latter alternative, however, will cause

SATMODEL's execution time to increase, and should be used only with considerable reserve, and only if the total number of optimization variables is within reason (see section 4.2 on optimization performance). We suggest that initial optimizations in a system design study be done with a single memory management algorithm, and that optimization over several algorithms be considered only in the later stages of design.

The value of the attribute is a character string of length between one and three, formed from the characters A, P, and W. No character may be repeated. Each character refers to one of the three memory management algorithms. If only a single character is given, the corresponding algorithm will always be used. If two or three are given, the optimizer will select the best algorithm.

The character A refers to a read always algorithm: each time the module is invoked, its code must be read in from backing store. This roughly corresponds to the way some overlay systems work in minicomputer operating systems.

The character P refers to the permanently resident algorithm: the module is always in main memory, so invoking it never incurs backing store overhead. This is the normal subsystem of usage in many operating systems and applications. Note that the model does not try to account for the time taken to initialize an application, a process which would of course involve loading main memory with all

the permanently resident modules.

The character W refers to the working set algorithm. A simple probabilistic model is used to calculate the probability that each module will already be in main memory and thus need not be loaded from backing store. The probability depends on the frequency of use of each module with respect to other modules, the total size of all modules being managed by the working set algorithm, and the amount of main store available. Details can be found in the First Annual Report . The intention is to model virtual memory environments such as those of the IBM 360/67, GE 645, PDP-11/45, and PDP-11/70.

All modules are allocated backing store space for permanent storage. Resident modules are of course assigned space in main memory of whichever computer they are assigned to. The collection of read always modules assigned to a particular computer take space equal in size to the largest of the modules. The collection of working set modules assigned to a particular computer requires at least as much space as the largest of the modules, but will use as much additional main memory as is available, up to the sum of the sizes of the modules.

Site: group MOD, entry Module Name, attribute MSIT. For each program module, the designer indicates one of three possibilities:

0. The module will be assigned to a computer site,

either host or satellite, by the optimization algorithm.

1. The module is permanently assigned to the host computer.
2. The module is permanently assigned to the satellite computer.

The integers 0, 1, and 2, corresponding to the three above options, are the three permissable values for MSIT.

Options 1 and 2 would be used for those modules which must be executed on a specific computer. Massive floating point computations, for instance, are normally done by the host computer. User interrupt handling, DPU program maintenance, and light pen correlation are usually done by the satellite computer. The number of modules assigned option 0, the so-called free modules, has a direct bearing on the execution time of SATMODEL (see section 4.2 on optimization performance). The most free modules we have ever used is about 20.

If the designer knows what division of labor he desires, then all modules would be fixed to the host or satellite computer. In this case SATMODEL would be used to find the appropriate satellite hardware configuration to use with the specified division of labor.

User Return Parameter Length: group MOD, entry Module Name, attribute RPL. This attribute is the average number

of bytes sent by the module to an output device (teletype, CRT, etc.) when the module terminates interaction. If the module cannot terminate an interaction, this attribute should be set to zero.

User_Return_Probability: group MOD, entry Module Name, attribute RPR. This attribute is the probability that this module will terminate the interaction by transfering control to the user, rather than transfering control to another module.

Within each Module Name entry, there is a group of zero or more File References which give information concerning module I/O activity. A more detailed description of how to describe files is in a later section. We specify the names of each file actually accessed by a module, a count of read accesses, and a count of write accesses.

File_Name: group MOD, entry Module Name, group FR, entry File Name. This is a unique character string name, of maximum length 80, defined by the designer.

Reads: group MOD, entry Module Name, group FR, entry File Name, attribute RA. This is the average number of times the module reads the file, per invocation of the module. Because we are specifying an average, the attribute's value need not be integer.

Writes: Group MOD, entry Module Name, group FR, entry File Name, attribute WA. This is the same as the Reads

attribute, but for Writes.

As will be seen in a later more detailed discussion, a file might be either a large aggregate of data items, or just a few specific pieces of data.

Within each Module Name entry, there is also a group of Transfers which give information concerning what other modules are transferred to by this module. We specify the name of each module actually transferred to, the probability of the transfer, and the amount of information transferred. A transfer can be a subroutine call or return, or even a simple jump.

To Module Name: group MOD, entry Module Name, group Transfers, entry to Module Name. The name of another module to which this module transfers control.

Probability: group MOD, entry Module Name, group Transfers, entry To Module Name, attribute PR. The probability that a transfer occurs from the entry Module Name to the entry To Module Name . Only transfers with non-zero probability need be given. The probabilities in the entry Module Name must sum to 1.0.

Parameter Length: group MOD, entry Module Name, group Transfers, entry to Module Name, attribute PL. The average number of bytes of parameters passed from Module Name to To Module Name when the tranfer of control occurs. If the two modules are on different computers, this is the length of

the message sent between the computers. The length is used to calculate the time needed to send an invocation message from a calling module on one computer to the called module on the other computer.

3.1.2 Description of Application Program Files

Just as the proposed (or actual) application program is subdivided into modules, so the application's data base is subdivided into files. Of course, there need not be any data base, in which case this group is empty. A data base, in the general sense, is a collection of one or more data items. A file may be as large as the data base, or as small as a single data item in the data base. It is possible that files contain duplicate information.

While one might conceivably divide a 10 item data base into 10 files, doing so would serve no useful purpose. Data bases are usually subdivided into a few files, with all items in a particular file often being related or similar to one another in specific ways. The number and nature of these files is established during the application design process. As with modules, the level of detail in the file structure is entirely at the designer's discretion. The more files used, however, the more time-consuming the optimization process. Each file has an entry in the FILE group:

File Name: group FILE, entry File Name. Each file is given a unique name of up to 80 characters. Each file has several attributes, all of which must be specified:

Buffer Size: group FILE, entry File Name, attribute BS. This is the size, in bytes, of the buffer needed to access the file. This would normally be the physical record length, or block size, of the file.

Total Size: group FILE, entry File Name, attribute TS. The maximum size, in bytes, of the file. This is the amount of bulk storage space which will be set aside for the file.

Access: group FILE, entry File Name, attribute ACC. This is a character string of length one which indicates how the file is accessed. An S means serial access, for which anticipatory input buffering is modeled if more than one buffer is avaialble for the file (see Buffer Number below). On output, with more than one buffer, overlapped output and program execution is modeled.

An R means a random access file, in which case no anticipatory buffering or overlapped output is attempted.

Buffer Management: group FILE, entry File Name, attribute BM. The designer can specify one of three algorithms with which the file's buffers are to be managed, or can have SATMODEL select the best algorithm. But just as with memory management for algorithms, the latter alternative will cause increased execution times for the

optimization algorithms.

The value of the attribute is a character string of length between one and three, formed from the characters A, P, and W. No character may be repeated. Each character refers to a specific algorithm. If a single character is given, the corresponding algorithm will always be used. If two or three are given, the optimizer will select the best algorithm.

The character A refers to an Allocate always algorithm. Buffer space is allocated before each I/O operation, then released afterwards. This means that main memory space is not permanently dedicated to buffers for the file.

The character P means Resident buffers. Buffers are permanently allocated, and do take main memory space. Most operating systems use this approach.

The character W means Working set buffers. A buffer will be in main memory unless "swapped out", or deallocated, in favor of the buffer for a more recently or frequently used file.

Buffer Number: group FILE, entry File Name, attribute BN. The maximum number of buffers (each of size given by the Buffer Size attribute) allocated for this file. This number is ignored if a buffer management algorithm of Allocate always is in use. There is no reason to specify more than one buffer for direct files; as their use is not

modeled.

Site: group file, entry File Name, attribute FSIT. This is interpreted in the same way as attribute MSIT for program modules. A value of 0 means the optimum site assignment is to be found; 1, the file will be at the host CPU; 2, the file will be at the satellite CPU.

3.2 Hardware-Software Tradeoffs

The hardware-software tradeoffs examined by SATMODEL have to do with the implementation of display operations. The applications file includes a description of all display operations used by the application, and some basic information associated with the operations. If the system designer is not interested in examining hardware-software tradeoffs, most of the information described here need not be given. Some, however, is needed if a flicker-free display capacity is specified. All information needed for this purpose is so indicated.

The display operation set has one entry per display operation. Each operation being considered for the hardware-software tradeoff would be included. Also, if flicker-free display capacity is of interest, a basic display operation such as "draw picture primitives" might be defined, although this could be instead represented as separate operations for each of the primitives such as points, lines, and text.

The order in which the hardware-software tradeoffs are considered is the order in which the display operations occur in this group. Since FED inserts only at the end of groups, changing the order requires deletion and re-entry of the data.

Operation Name: group DOS, entry Operation Name. The name is a string of up to 80 characters. The name must match a name in the instruction set list of each Remote DPU in the hardware file. Three attributes are given with each entry in the DOS.

Parameter Length: group DOS, entry Operation Name, attribute OPL. This is the average number of bytes of information required by the display operation each time it is invoked. For instance, a simple draw instruction in a 2-D DPU would need 2 bytes for an x coordinate, and 2 bytes for y. A rotate instruction requires a rotation matrix plus all the coordinates which will be rotated. A draw text instruction requires one byte per character.

If the application program module which invokes the display operation is executing on the host, this is the number of bytes which will be sent to the satellite to cause the operation to be performed.

Site: group DOS, entry Operation Name, attribute OSIT. This attribute's value is currently ignored by SATMODEL, having been included for future expansions, a value must be specified. The implementation of a display operation, be it in hardare or software, is forced to be at the satellite. The legal values of the attribute are 0, 1, or 2. Their interpretations are the same as for module and file site attributes.

Memory Management: group DOS, entry Operation Name, attribute OMEM. This attribute's value is ignored by SATMODEL, but a value must be specified. The program module used to implement the display operation is assumed to be permanently resident in main memory. The attribute value must be a character string of length 1 to 3 containing no more than one occurence of each of the letters R, A, and W. The character's interpretations are the same as for module memory management.

Display Operations: group MOD, entry Module Name, Group DO. Within each Module Name entry, there is also a group of Display Operations. One purpose of this group is to allow the optimizer to account for the degradation of satellite CPU performance caused by asynchronous user interactions. For instance, in a 3-D drawing application, the user might have a joystick to control the viewing angle. Moving the joystick causes the object displayed on the screen to rotate, whether or not other processing is occurring on the satellite. If other processing is indeed taking place, we assume that it has lower priority than the computations required to effect the rotation. The question of hardware-software tradeoffs comes in here: if the rotation is carried out in hardware, there will be less CPU performance degradation than if software rotation is necessary. Some applications might not require or allow this sort of concurrency, in which case the group of display operations would not be used in this way. The display operation group

is also used to account for CPU performance degradation caused by performing some display operations in CPU software rather than in DPU hardware, as part of the regular refresh process. Performing subpicture calls with CPU aid is an example.

Another purpose of the display operations is to directly account for the effect of hardware-software tradeoffs on response time. For all display operations which might be implemented in either hardware or software and which the module directly calls, the designer specifies how many times such operations are called.

Operation Name: group MOD, entry Module Name, group DO, entry Operation Name. This is the character-string name (maximum length 80) of the display operation. The name must match a name in the Display Operation Set group, described above. This entry has just one attribute:

Operation Count : group MOD, entry Module Name, group DO, entry Operation Name, attribute DOC. This count is the sum of three quantities. The first is the number of times this display operation is invoked asynchronously by user actions each time the module is active; the second, the number of times this display operation is invoked asynchronously as part of the DPU refresh cycle each time the module is active; the third, the number of times this display operation is directly called by the module, each time the module is active.

3.3 Selective Specifications

3.3.1 Flicker-free Display Requirements

The application designer can describe how complex the pictures to be drawn are, and how fast the refresh rate must be. SATMODEL then discards those DPUs which are too slow to display the pictures flicker-free. The description is of individual pictures. The designer should describe the largest pictures, since they take the longest to display.

For purposes of conceptualization, each picture is described as a series of subpictures, each with one or more occurences. Of course a picture may contain just one occurrence of one subpicture.

Flicker Tolerance: group REQ, attribute FL. This is the minimum refresh rate, in frames per second, needed to draw a flicker-free picture with the phosphor selected for the application.

Display Usage: group REQ, group DU. This is a group of pictures. Each picture has a name, and one or more subpictures.

Picture Name: group REQ, group DU, entry Picture Name. A string of up to 8 characters.

Subpicture Name: group REQ, group DU, entry Picture Name, entry Subpicture Name. A string of up to 80

characters.

Multiplicity: group REQ, group DU, entry Picture Name, entry Subpicture Name, attribute SMUL. The number of times this subpicture occurs in the picture being described

Operations: group REQ, group DU, entry Picture Name, entry Subpicture Name, group POP. The display operations used by the subpicture being described.

Operation Name: group REQ, group DU, entry Picture Name, entry Subpicture Name, group POP, entry Operation Name. The name must also occur in the instruction set description for each DPU in the hardware file.

Multiplicity: group REQ, group DU, entry Picture Name, entry Subpicture NAme, group POP, attribute POM. The number of times the operation occurs in the subpictures being described.

The time taken to draw a particular picture is the sum of the times taken to draw each instance of each subpicture. The maximum time to draw any of the picutures is compared to the refresh rate to determine whether the DPU is acceptable.

The time taken to draw a subpicture is the sum of the times needed to execute each invocation of each display operation. These times are taken from the hardware file. If a display operation has two implementations (hardware and software), the lesser of the two times is used here.

3.3.2 General Selective Specification

Each hardware subsystem (Data Link, Remote Fast Store, Remote Bulk Store, Satellite CPU, DPU) in the hardware file can have an associated set of supplemental characteristics, for use in selective specification. These characteristics might be resolution and spot size for the DPU, error rate for the Data Link, etc.

This optional section of the application file is used to give the minimum acceptable values for each such supplemental characteristic. A subsystem whose value is greater than or equal to the mimimum will be accepted and considered in the optimization process; else, it is rejected and not considered further. Also, if the supplemental characteristic is missing from a subsystem's description, the subsytem is discarded.

Since the acceptance test is "greater than or equal to", supplemental attributes like spot size (where a smaller value is better than a larger value) must be specified as an inverse, i.e., 1/inches rather than inches. Also, non-numeric attributes such as half-duplex and full-duplex must be numerically coded in order of increasing preference, i.e., 1 = half-duplex, 2 = full-duplex.

Selective specification can be done only with supplemental characteristics of the subsystems in the hardware file. Thus if only data links with speed greater

than same mimimum are to be considered, the speed must be duplicated as a supplemental characteristic, even though it also appears as a data link attribute.

The groups DLS, RFSS, RBSS, CPUS, and DPUS are used to give the supplemental characteristics names (as an entry) and minimum value (as an attribute under the name).

3.4 User characteristics

Two user characteristics are of importance: the average think time (time between interactions) and the probability distribution of initial program module activation caused by user interactions. These characteristics must be included in the application program description.

Userthink Time: group USER, attribute UTT. The user think time, in seconds.

User Module References: group USEP, group UMR. A list of all program modules which might be directly activated by a user action. These are the modules which actually wait for a user action (pen hit, character string input, etc.) In many applications, there is only one such module.

Referenced Module Name: group USEP, group UMR, entry Referenced Module Name. The name of such a module. The name must match that of application program module.

Probability: group USEP, group UMP, attribute UPP. The probability that this module is the first module activated by a user action. The probabilities for all modules in group UMP must sum to 1.0.

Parameter Length: group USEP, group UMR, attribute UPL. The number of bytes and information passed by the system software to the module. If the module is host-resident then this is the length of the message sent accross

the data link to the module.

3.5 Example

Figure 3.2 is a block diagram of the modules and files used to model the application program of figure 3.1.

Figure 3.3 is the listing of an application file for the program shown in figure 3.2. It is designed for a basic optimization, without regard for selective specification or hardware-software tradeoffs.

Figure 3.4 is a copy of the application file in figure 3.3 which has been modified to study hardware-software tradeoffs and to use selective specifications. It has been agmented (in places marked by an '*') to allow hardware-software tradeoffs for subpicture subroutine jumps, rotation, and basic operations. It also provides information for selective specifications on the basis of flicker-free display capacity and the supplemented characteristics for each of the subsystems. The files shown in figures 3.3 and 3.4 are compatible with the hardware file shown in figure 2.1.

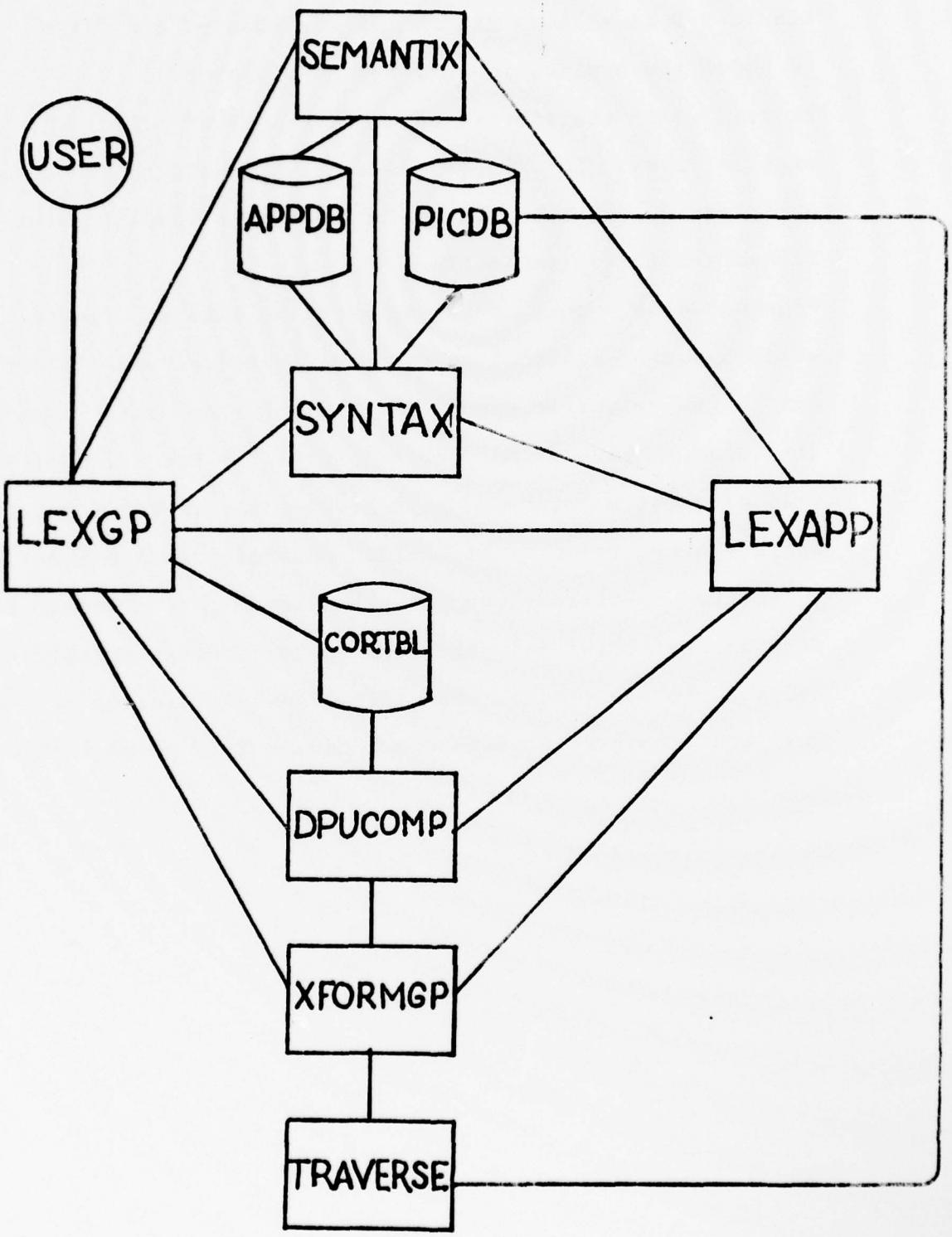


Figure 3.2 The Modules and Files of an Application Program

```
application file(af)
   display operation set(dos)
   requirements(req)
      flicker tolerance(fl)=    0.000000
      display usage(du)
   data link specifications(dls)
   remote fast store specifications(rfss)
   remote bulk store specifications(rbss)
   cpu specifications(cpus)
   dpu specifications(dpus)
   modules(mod)
      syntax
         size(msiz)=    10240
         instructions(mins)=  100.000000
         memory management(mmem)=w
         site(msit)=        0
         user return parameter length(rpl)=        0
         user return probability(rpr)=    0.000000
         file references(fr)
            appdb
               reads(ra)=    0.200000
               writes(wa)=    0.200000
            picdb
               reads(ra)=    0.500000
               writes(wa)=    0.300000
         transfers(tr)
            lexgp
               probability(pr)=    0.250000
               parameter length(pl)=      250
            semantix
               probability(pr)=    0.200000
               parameter length(pl)=     2000
            lexapp
               probability(pr)=    0.450000
               parameter length(pl)=      250
         display operations(do)
      lexgp
         size(msiz)=     1000
         instructions(mins)=  100.000000
         memory management(mmem)=w
         site(msit)=        2
         user return parameter length(rpl)=       10
         user return probability(rpr)=    0.500000
         file references(fr)
            cortbl
               reads(ra)=    0.500000
               writes(wa)=    0.100000
```

Figure 3.3 A Basic Application File

```
      transfers(tr)
         semantix
            probability(pr)=    0.050000
            parameter length(pl)=      250
         syntax
            probability(pr)=    0.100000
            parameter length(pl)=      250
         lexapp
            probability(pr)=    0.200000
            parameter length(pl)=      100
         dpucomp
            probability(pr)=    0.100000
            parameter length(pl)=       50
         xformgp
            probability(pr)=    0.050000
            parameter length(pl)=      500
      display operations(do)
   xformgp
      size(msiz)=    30720
      instructions(mins)=  300.000000
      memory management(mmem)=w
      site(msit)=        2
      user return parameter length(rpl)=        0
      user return probability(rpr)=    0.000000
      file references(fr)
      transfers(tr)
         lexapp
            probability(pr)=    0.100000
            parameter length(pl)=       50
         lexgp
            probability(pr)=    0.100000
            parameter length(pl)=       50
         dpucomp
            probability(pr)=    0.300000
            parameter length(pl)=      100
         traverse
            probability(pr)=    0. 00000
           parameter length(pl)=      500
      display operations(do)
   dpucomp
      size(msiz)=    18432
      instructions(mins)=  300.000000
      memory management(mmem)=w
      site(msit)=        2
      user return parameter length(rpl)=        0
      user return probability(rpr)=    0.000000
      file references(fr)
         corthl
            reads(ra)=    1.500000
            writes(wa)=    2.500000
```

Figure 3.3 (continued)

```
      transfers(tr)
         lexapp
          probability(pr)=     0.300000
            parameter length(pl)=       100
         lexgp
            probability(pr)=     0.400000
            parameter length(pl)=       100
         xformgp
            probability(pr)=     0.300000
            parameter length(pl)=       100
      display operations(do)
   traverse
      size(msiz)=      5120
      instructions(mins)=  300.000000
      memory management(mmem)=w
      site(msit)=         0
      user return parameter length(rpl)=         0
      user return probability(rpr)=     0.000000
      file references(fr)
         picdb
            reads(ra)=   10.000000
            writes(wa)=    0.010000
      transfers(tr)
         xformgp
            probability(pr)=     1.000000
            parameter length(pl)=        50
      display operations(do)
   lexapp
      size(msiz)=      5120
      instructions(mins)=   50.000000
      memory management(mmem)=w
      site(msit)=         0
      user return parameter length(rpl)=         0
      user return probability(rpr)=     0.000000
      file references(fr)
      transfers(tr)
         semantix
            probability(pr)=     0.200000
            parameter length(pl)=        50
         syntax
            probability(pr)=     0.200000
            parameter length(pl)=        50
         lexgp
            probability(pr)=     0.400000
            parameter length(pl)=        10
         dpucomp
            probability(pr)=     0.100000
            parameter length(pl)=        10
         xformgp
            probability(pr)=     0.100000
            parameter length(pl)=        20
      display operations(do)
```

Figure 3.3 (continued)

```
      semantix
         size(msiz)=     51200
         instructions(mins)=  500.000000
         memory management(mmgt)=v
         site(msit)=         0
         user return parameter length(rpl)=         0
         user return probability(rpr)=     0.000000
         file references(fr)
            appdb
               reads(ra)=     0.700000
               writes(wa)=    0.500000
            picdb
               reads(ra)=     0.400000
               writes(wa)=    0.200000
         transfers(tr)
            syntax
               probability(pr)=     0.750000
               parameter length(pl)=       200
            lexgp
               probability(pr)=     0.100000
               parameter length(pl)=        50
            lexapp
               probability(pr)=     0.150000
               parameter length(pl)=        50
         display operations(do)
   files(file)
      appdb
         buffer size(bs)=     5120
         number of buffers(bn)=         1
         total size(ts)=   102400
         access(acc)=r
         buffer management(bm)=a
         site(fsit)=         1
      picdb
         buffer size(bs)=     5120
         number of buffers(bn)=         1
         total size(ts)=    51200
         access(acc)=r
         buffer management(bm)=v
         site(fsit)=         2
      cortbl
         buffer size(bs)=     5120
         number of buffers(bn)=         1
         total size(ts)=    20480
         access(acc)=r
         buffer management(bm)=v
         site(fsit)=         2
   user(user)
      user think time(utt)=     5.700000
      user module references(umr)
         lexgp
            probability(upr)=     1.000000
            parameter length(upl)=        80
```

Figure 3.3 (continued)

```
application file(af)
   modules(mod)
      syntax
         size(msiz)=    10240
         instructions(mins)=  100.000000
         memory management(mmem)=w
         site(msit)=        1
         user return parameter length(rpl)=        0
         user return probability(rpr)=    0.000000
         file references(fr)
            appdb
               reads(ra)=    0.200000
               writes(wa)=    0.200000
            picdb
               reads(ra)=    0.500000
               writes(wa)=    0.500000
         transfers(tr)
            lexgp
              probability(pr)=    0.250000
               parameter length(pl)=      250
            semantix
               probability(pr)=    0.200000
               parameter length(pl)=     2000
            lexapp
               probability(pr)=    0.450000
               parameter length(pl)=      250
         display operations(do)
      lexgp
         size(msiz)=     1000
         instructions(mins)=  100.000000
         memory management(mmem)=w
         site(msit)=        2
         user return parameter length(rpl)=       10
         user return probability(rpr)=    0.500000
         file references(fr)
            cortbl
               reads(ra)=    0.500000
               writes(wa)=    0.100000
         transfers(tr)
            semantix
               probability(pr)=    0.050000
               parameter length(pl)=      250
            syntax
               probability(pr)=    0.100000
               parameter length(pl)=      250
            lexapp
               probability(pr)=    0.200000
               parameter length(pl)=      100
            dpucomp
               probability(pr)=    0.100000
               parameter length(pl)=       50
            xformgp
               probability(pr)=    0.050000
               parameter length(pl)=      500
         display operations(do)
```

Figure 3.4 An Application File with Hardware-Software Tradeoffs

```
xformgp
   size(msiz)=    30720
   instructions(mins)=  300.000000
   memory management(mmem)=w
   site(msit)=        2
   user return parameter length(rpl)=        0
   user return probability(rpr)=    0.000000
   file references(fr)
   transfers(tr)
      lexapp
         probability(pr)=    0.100000
         parameter length(pl)=       50
      lexgp
         probability(pr)=    0.100000
         parameter length(pl)=       50
      dpucomp
         probability(pr)=    0.300000
         parameter length(pl)=      100
      traverse
        probability(pr)=    0.500000
         parameter length(pl)=      500
   display operations(do)
dpucomp
   size(msiz)=    18432
   instructions(mins)=  300.000000
   memory management(mmem)=w
   site(msit)=        2
   user return parameter length(rpl)=        0
   user return probability(rpr)=    0.000000
   file references(fr)
      cortbl
         reads(ra)=    1.500000
         writes(wa)=    2.500000
   transfers(tr)
      lexapp
         probability(pr)=    0.500000
         parameter length(pl)=      100
      lexgp
         probability(pr)=    0.400000
         parameter length(pl)=      100
      xformgp
         probability(pr)=    0.300000
         parameter length(pl)=      100
   display operations(do)
*     Rotation
*        operation count(doc)=        3
traverse
   size(msiz)=     5120
   instructions(mins)=  300.000000
   memory management(mmem)=w
   site(msit)=        2
   user return parameter length(rpl)=        0
   user return probability(rpr)=    0.000000
   file references(fr)
      picdb
         reads(ra)=   10.000000
         writes(wa)=    0.010000
```

Figure 3.4 (continued)

```
files(file)
   appdb
      buffer size(bs)=      5120
      number of buffers(bn)=          1
      total size(ts)=    102400
      access(acc)=r
      buffer management(bm)=a
      site(fsit)=          1
   picdb
    buffer size(bs)=      5120
      number of buffers(bn)=          1
      total size(ts)=     51200
      access(acc)=r
      buffer management(bm)=v
      site(fsit)=          2
   cortbl
      buffer size(bs)=      5120
      number of buffers(bn)=          1
      total size(ts)=     20480
      access(acc)=r
      buffer management(bm)=v
      site(fsit)=          2
display operation set(dos)
*    Rotation
*       parameter length(opl)=          1
*       site(osit)=          2
*       memory management(omm)=r
*       hardware-software option(ho)=hs
*    Basic Operation Set
*       parameter length(opl)=          1
*       site(osit)=          2
*       memory management(omm)=r
*       hardware-software option(ho)=hs
*    Subroutine Jump
*       parameter length(opl)=          1
*       site(osit)=          2
*       memory management(omm)=r
*       hardware-software option(ho)=hs
user(user)
   user think time(utt)=     5.700000
   user module references(umr)
      lexgp
         probability(upr)=     1.000000
         parameter length(upl)=         80
requirements(req)
*    flicker tolerance(fl)=     0.030000
*    display usage(du)
*       Power Amplifier
*          Transistor
*             multiplicity(smul)=          2
*             operations(pop)
*                Basic Operation Set
*                   multiplicity(pom)=   120.000000
*                Subroutine Jump
*                   multiplicity(pom)=    10.000000
```

Figure 3.4 (continued)

```
   transfers(tr)
      xformgp
         probability(pr)=    1.000000
        parameter length(pl)=       50
   display operations(do)
lexapp
   size(msiz)=     5120
   instructions(mins)=   50.000000
  memory management(mmgr)=w
   site(msit)=        1
   user return parameter length(rpl)=        0
  user return probability(rpr)=    0.000000
   file references(fr)
   transfers(tr)
      semantix
         probability(pr)=    0.200000
         parameter length(pl)=       50
      syntax
         probability(pr)=    0.200000
         parameter length(pl)=       50
      lexgp
        probability(pr)=    0.400000
         parameter length(pl)=       10
      dpucomp
         probability(pr)=    0.100000
        parameter length(pl)=       10
      xformgp
         probability(pr)=    0.100000
         parameter length(pl)=       20
   display operations(do)
semantix
   size(msiz)=    51200
   instructions(mins)=  500.000000
   memory management(mmgr)=w
   site(msit)=        1
   user return parameter length(rpl)=        0
   user return probability(rpr)=    0.000000
   file references(fr)
      appdb
         reads(ra)=    0.700000
         writes(wa)=    0.500000
      picdb
         reads(ra)=    0.400000
         writes(wa)=    0.200000
   transfers(tr)
      syntax
         probability(pr)=    0.750000
         parameter length(pl)=      200
      lexgp
         probability(pr)=    0.100000
         parameter length(pl)=       50
      lexapp
         probability(pr)=    0.150000
         parameter length(pl)=       50
   display operations(do)
```

Figure 3.4 (continued)

```
*          Resistor
*             multiplicity(smul)=         5
*             operations(pop)
*                Basic Operation Set
*                   multiplicity(pom)=  100.000000
*                Subroutine Jump
*                   multiplicity(pom)=   10.000000
*                Rotation
*                   multiplicity(pom)=    1.000000
*          Capacitor
*             multiplicity(smul)=         2
*             operations(pop)
*                Basic Operation Set
*                   multiplicity(pom)=   40.000000
*                Subroutine Jump
*                   multiplicity(pom)=   10.000000
*                Rotation
*                   multiplicity(pom)=    1.000000
*          Transformer
*             multiplicity(smul)=         2
*             operations(pop)
*                Basic Operation Set
*                   multiplicity(pom)=   18.000000
*                Subroutine Jump
*                   multiplicity(pom)=    1.000000
*          Diode
*             multiplicity(smul)=         1
*             operations(pop)
*                Basic Operation Set
*                   multiplicity(pom)=   70.000000
*                Subroutine Jump
*                   multiplicity(pom)=   10.000000
*                Rotation
*                   multiplicity(pom)=    5.000000
   remote fast store specifications(rfss)
*     Cycle Time
*        value(val)=      1000
   remote bulk store specifications(rbss)
*     Size
*        value(val)= -2000000
   cpu specifications(cpus)
*     Access Time
*        value(val)=      1000
   dpu specifications(dpus)
*     resolution
*        value(val)=        50
*     colors
*        value(val)=         1
   data link specifications(dls)
*     Distance
*        value(val)=         1
```

Figure 3.4 (continued)

CHAPTER 4: OPTIMIZATION

The optimization process has four major inputs:

- a hardware file, describing the host computer and the subsystems' price and performance
- an application file, describing the interactive graphics application program for which the satellite is being designed
- the maximum allowable cost of the satellite system
- the number of interactive display consoles connected to the satellite computer.

The optimization output goes to a file, and is a description of the satellite system hardware and the software divisions of labor which provides the fastest response time and costs no more than the specified maximum number of dollars. Since response time is increased by selecting faster, more expensive subsystems, the optimum satellite system in cost will be close to the maximum cost. Since we will deal with a fairly small number of subsystems, with discrete costs, it would be unusual for the satellite's cost to exactly equal the maximum cost.

The input and output files and other optimization parameters are specified to the OPT mode of SATMODEL. The optimization occurs in several steps: amalgamating the hardware and application program descriptions, and then

searching for the optimum system.

The two files are first read and validated. Subsystems not meeting the selective specifications are discarded. If a flicker rate limitation is specified in the application file, then DPU's which cannot meet the requirement are discarded. Then the DPU's are paired with compatible CPU's chosen from the hardware file. At this point the entire input data is checked for completeness and consistency. Finally, the actual optimization is performed.

4.1 General Use

Within this general optimization framework, the optimizer can be used in several ways.

Division of Labor Only

A hardware file, containing only one of each subsystem, is created. The subsystems are those of the satellite graphics system for which an optimum division of labor is desired. The application file describes the appropriate application. The SITE attribute for the modules and files which can be resident at either the host or satellite is set to 0. For all others, the SITE attribute is set to represent the permanent assignment. The maximum cost is set to any value greater than or equal to the sum of the individual subsystems' costs.

Optimum Satellite System Only

A full hardware file is used, with several different entries for each type of subsystem. In the applications file, all SITE attributes are set to one or two, giving a fixed division of labor. Memory management attributes would also be assigned a single value, so they don't enter into the optimization process either. Since any division of labor implies minimum memory requirements at the satellite, some satellite main and bulk memory subsystems may not be considered in the optimization.

The desired maximum cost is given, as usual.

Cost vs. Response Time Curves

Curves for cost versus minimum response time, such as those in The Second Annual Report, are very useful in finding when the returns on increased investment begin to decrease, or to find the cost of attaining a specified response time. Data points on the curve are found by making multiple optimization runs changing only the maximum cost between runs. One should start with a maximum cost equal to the sum of the costs of each of the lowest-priced subsystems, and end with the sum of the costs of each of the highest-priced subsystems. From 5 to 10 intermediate costs are generally sufficient to give a reasonably useful curve.

4.2 Optimization Speed

With the TUCC IBM 370/165, 5 to 10 minutes of CPU time were needed to do a hardware optimization with about 4 or 5 different choices of each of the four subsystems and no more than 20 unassigned modules or files in the application description. We cannot directly relate this time to an equivalent number of GE645 minutes. However, since MULTICS is a multi-user, interactive system, as much as 20 or 30 actual minutes may be needed to obtain 1 minute of CPU time. Assuming a 1 to 1 speed ratio, the 10 minutes of CPU time could require as much as five hours of elapsed time. We thus suggest overnight "batch" runs for optimization of this scope. Some smaller optimization will doubtless be feasible to do interactively.

If the number of consoles at the satellite is more than one, optimization times are greater than in the single-user case, since a network of queues is solved using an iterative algorithm, RQA. Thus multi-console systems should be analyzed sparingly, and with as few subsystem choices and unbound modules and files as possible. The maximum allowable number of consoles is about five.