COMPUTER AIDED SIMULATION FOR COMPUTER SYSTEM STUDIES

George K.Hutchinson

ABSTRACT

This paper introduces a simulation system, CAPS, which interacts with an analyst via an on-line dialog and produces a simulation program that is logically consistent and executes on first submittal. CAPS is based upon the use of activity cycles for system decomposition. Activity cycles are discussed and the system demonstrated by the simulation of a computing system. The major advantages of CAPS are the speed with which models can be implemented and its ease of use which permits non-programmers to develop sophisticated models. For example, the demonstration model of an interactive computer system, from the start of the CAPS dialog to simulation output, was 23 minutes at a cost of $3.29.

INTRODUCTION

Simulation is now well established as a problem solving tool, yet its use is still severely limited in comparison with the set of problems to which it could contribute solutions. The major reasons for its limited use may be the cost of simulations, the long lead times usually involved, and/or the limited number of people with the ability to use simulation languages. A. T. Clementson [1], recognizing these problems, applied the principles of computer aided design with Computer Aided Programming for Simulation (CAPS) as the result. CAPS interacts with the user, who need have no programming experience, to define the user's model. When this is accomplished, CAPS writes a simulation program which is logically consistent and will execute on the first run. This results in a substantial reduction in the time span from problem definition to simulation output and increases significantly the number of people who can use simulation as a problem solving tool.

CAPS is based upon the use of activity cycles as the means of decomposing the system under study. Activity cycles have long been used in England for both systems analysis and simulation. Hills [5] developed the HOCUS simulation language using activity cycles. HOCUS had several of the characteristics of CAPS but developed its program directly in machine language which limited its use. In contrast, CAPS generated programs are written in Extended Control Simulation Language - ECSL, the most popular simulation language in England. This paper will show the ease with which simulations can be performed with CAPS.

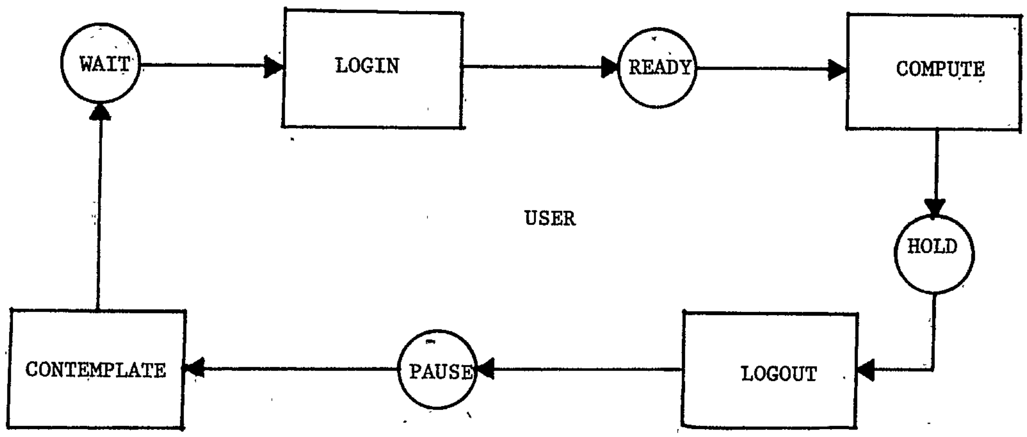
The first section contains an introduction to activity cycles and the CAPS dialog, using the simulation of a computing system as an example. This example is expanded to demonstrate the method by which a model can be expanded to incorporate more detail. The appendix contains the actual CAPS dialog, a listing of the CAPS generated code, execution results, and analysis. Interested readers should gain a basic knowledge of CAPS and activity cycles from reading the paper and see the ease and speed with which complex systems can be modeled.

AN INTRODUCTION TO CAPS

Any approach to system analysis or simulation of complex systems requires that the system be broken down into simpler and smaller subsystems for ease of manipulation and understanding. This process is decomposition and activity cycle analysis is one of the methods used to achieve this goal. For the purposes of this paper, a system is considered to be composed of entities, things which we wish to talk about and whose behavior we wish to describe as time advances. In a factory, the entities might be men, machines and jobs. In a store, they might be customers, clerks, and helpers. A computer system might include a CPU, disk, and terminals. These entities may have attributes which distinguish and describe them. Customers might have budgets and number of items. Clerks might have check-out rates and skill levels. Helpers could be described by pay and performance rates, the disk by its read/write speed.

The basic step in decomposing a system under study is to identify the entities of interest and group them into classes having similar or identical behavior patterns. For instance, a user of a computing system might have the activity cycle shown in Figure 1, where the active states are shown as boxes and the idle states, or queues, as circles. The queue, PAUSE, might have zero duration and go immediately from activity LOGOUT to CONTEMPLATE. From the diagram it appears that the user will immediately attempt to perform activity LOGIN after CONTEMPLATE is finished. Actually this merely reflects the fact that the diagram must be drawn so

FIGURE 1.
User Activity Cycle Diagram



as to close the cycle for each entity, a requirement of CAPS. The implications of this will be discussed later.

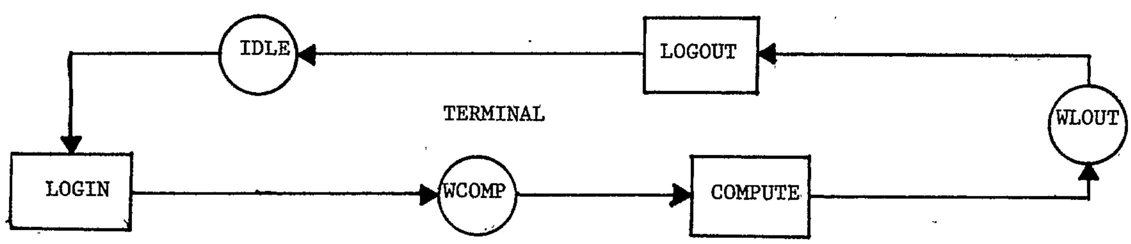
In most systems of interest, important entities will usually spend some time in the queues, because the activity for which they are queued requires more than one entity before it can be undertaken. These activities are known as cooperative activities. For instance, activity LOGIN might require a terminal. If the terminal were also required for COMPUTE and LOGOUT, its activity cycle diagram would be shown in Figure 2. The basic rule for

CONTEMPLATE once the entities for LOGIN become available; thus starting LOGIN has as its logical consequence the starting of CONTEMPLATE, albeit at a later point in time.

Since this is not the way our system operates, i.e., there is a computer, all entities of importance must be included. In Figure 4, the computer cycle is added. Note that activity COMPUTE is no longer bound.

One of the more interesting aspects of activity cycles is their power in portraying the logical

FIGURE 2.
Terminal Activity Cycle Diagram



cooperative activities is that each of the entities required by the activity must be in its immediate predecessor queue before the activity can begin. Thus LOGIN cannot take place until a terminal is in queue IDLE and a user is in queue WAIT.

The other major class of activities is the bound activity. A bound activity is one such that all of the entities required come to it directly from a single predecessor activity. For instance, if the activity cycles for users and terminals are combined, as in Figure 3, activities COMPUTE, LOGOUT, and CONTEMPLATE are bound activities. CONTEMPLATE is bound to LOGOUT and the other two to LOGIN. As drawn, no additional resources are required for

relationships of the system under study. The activity cycle diagram in Figure 4 is independent of 1) the number of terminals, 2) the computers, and 3) the number of users. The diagram is applicable to all computing systems where a user arrives, waits for a terminal, logs in, uses the computer, logs out, and leaves — to restart the cycle later (an issue yet to be considered). The complexities associated with the quantities of the entities interacting disappear and the analyst can concentrate on their behavior.

Another of the virtues of activity cycle diagrams is that they encourage one to start with macro models of a system and expand those activities which

FIGURE 3.

Combined User and Terminal Activity Cycles

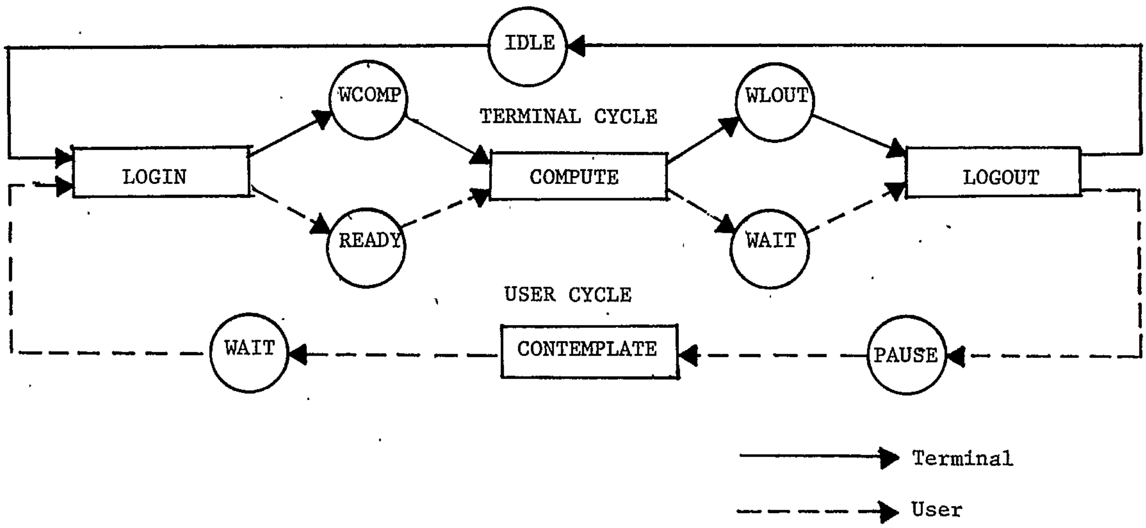
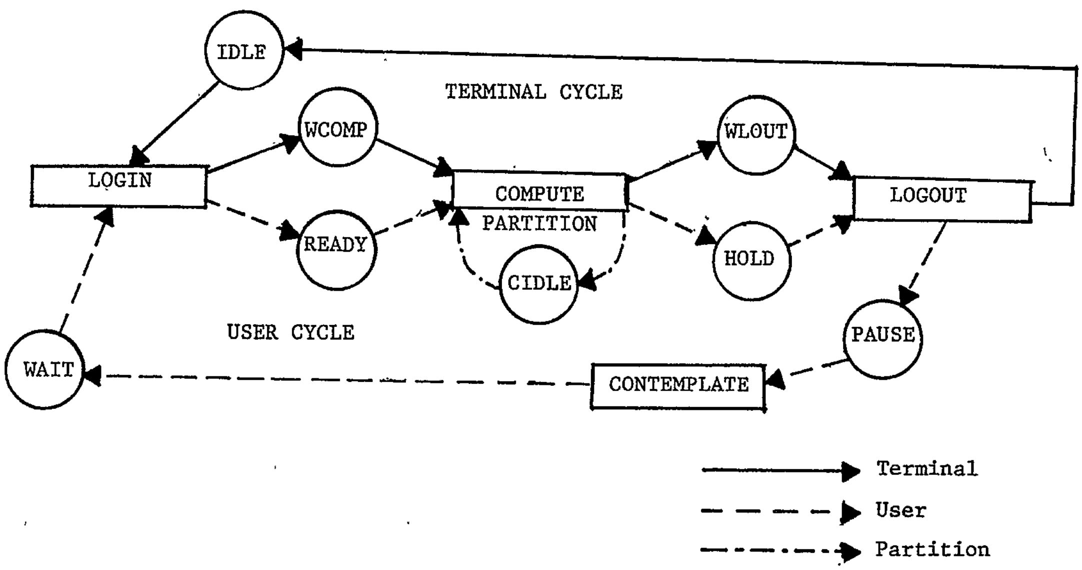


FIGURE 4.

Activity Cycle Diagram with Computer



early simulations show are most sensitive, rather than including all model elements at the same level of detail. This procedure tends to produce better models for the same effort.

In the computing system, one might argue that no user would ever have the exclusive use of the computer, as shown. One answer could be to think of the computer as being a partition which the user would occupy for the appropriate time span, rather than as the central processor. As an alternative, consider a situation where COMPUTE is expanded as follows: once a partition is obtained there is a period of main frame usage followed by I/O activity requiring a disk on a dedicated channel; after which, either another cycle of main frame - I/O or the completion of COMPUTE. The original COMPUTE is shown in Figure 5 and the more detailed cycle diagram in Figure 6. To conserve readability, the immediate queues have not been labeled. The

FIGURE 5.

Original COMPUTE Activity

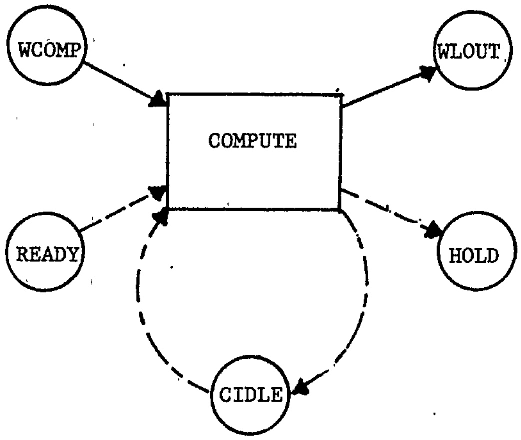
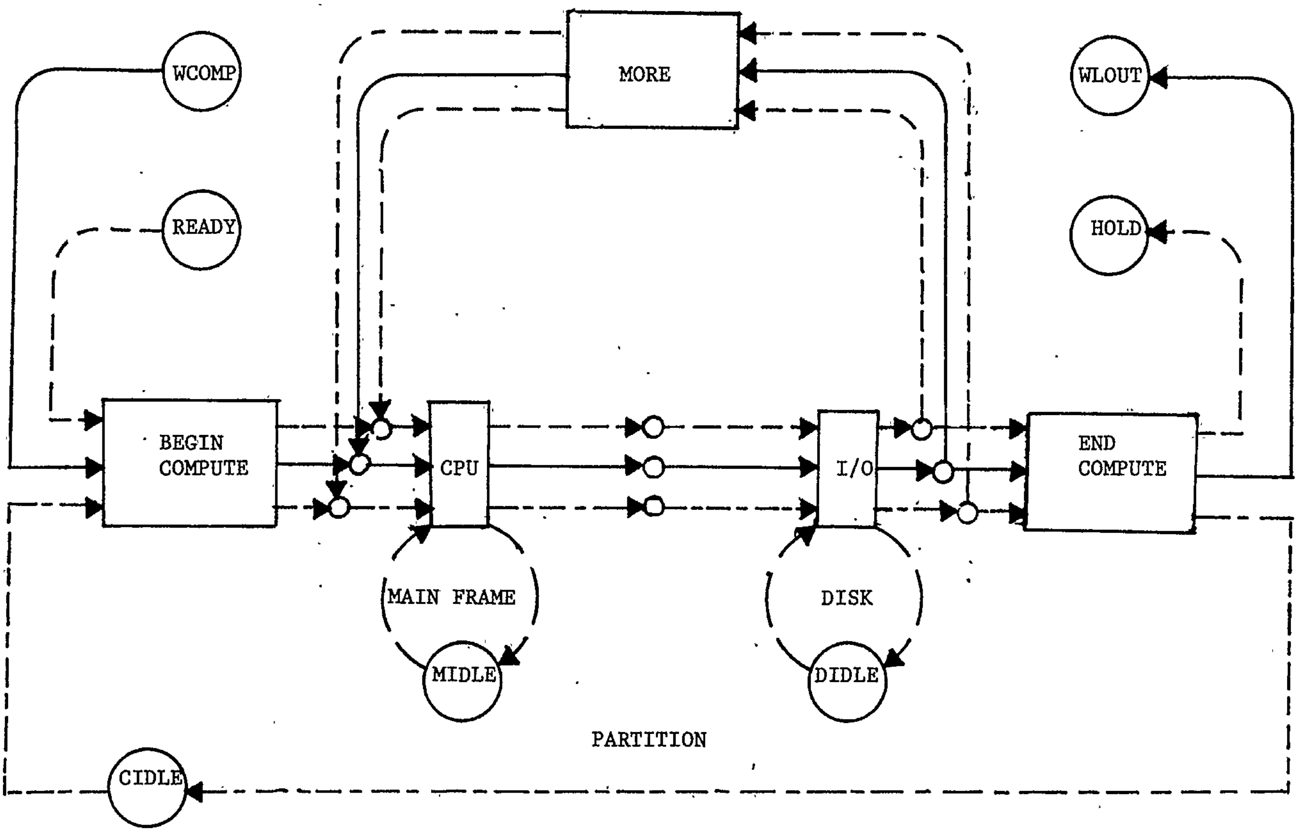


FIGURE 6.

Expanded COMPUTE Activity

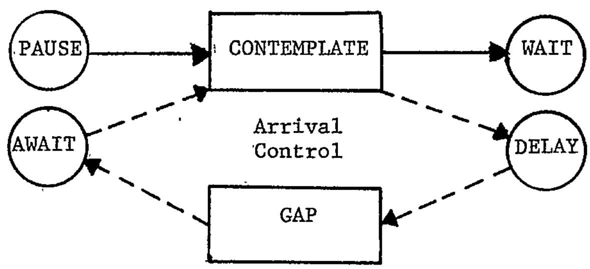


activity cycle diagram is still independent of the number of each of the resources involved. The queues for interface with the remaining elements of the model are the same in Figures 5 and 6, indicating the ease with which a model can be expanded to encompass more detail. In fact, one could expand any of the activities in Figure 6 with the same procedure. The remainder of this paper demonstrates the use of CAPS by modifying the model portrayed in Figure 4.

The User Cycle indicates that, upon completion of LOGOUT, the bound activity CONTEMPLATE merely delays the arrival of the user at WAIT where he will undertake LOGIN when a terminal is available. Essentially the prime factors of interest in the system are the arrival rate of users and their service. In reality, once LOGOUT is complete, the user leaves the system. CAPS requires that all cycles be closed, so CONTEMPLATE is used as a path from PAUSE to WAIT. As drawn, it is a bound activity which one might wish to change to control user arrival patterns. Making the normal assumption of negative expotential arrivals, an additional entity, Arrival Control, is added as shown in Figure 7. CONTEMPLATE is now a cooperative

FIGURE 7.

Arrival Control



activity requiring a user and an arrival control, the latter being a logical rather than physical entity. By choosing the proper distribution of times for GAP, the interarrival times, the desired distribution of user arrivals can be achieved.

To complete the activity cycle diagram the usual procedure is to add the number available for each entity and the duration time of each activity. This information is required by CAPS and its inclusion on the activity cycle diagram aids the analyst during the CAPS dialog. The completed diagram is given in Figure 8; with 15 users, 6 terminals, 4 computer partitions, and 1 arrival control. The durations of activities are shown as "D=". The durations may be any arithmetic expression and/or distribution, including the sampling of histograms. The names chosen for queues and activities are arbitrary unique strings of up to 6 alpha characters (additional characters may be used but are ignored by CAPS). The diagram in Figure 8 is now complete and portrays the basic information required by CAPS. It is the basis for both the dialog that follows and the actual system interaction shown in the Appendix.

CAPS DIALOGS

The basic input to CAPS is the topology of the activity cycle diagram. The cycles are specified, upon request by CAPS, by giving for each entity, the alternative queues--preceded by a Q--and activities--preceded by an A. CAPS performs many logic and consistency tests as the user supplies these cycles pointing out the consequences of the user's model and any inconsistencies (see Reference 6 for details). In fact, CAPS will not allow a user to proceed until a logically consistent model has been specified. Users are often surprised by CAPS ability to point out the logical consequences of their input, such as "LOGOUT IS A BOUND ACTIVITY (IT WILL START IMMEDIATELY UPON COMPLETION OF THE PRECEDING ACTIVITY)" or "NO MORE THAN 7 OF THE 15 USERS CAN BE ACTIVE AT ONE TIME," or most devastating, "YOUR PROBLEM DOES NOT REQUIRE SIMULATION, THE STATIC SOLUTION IS ---".

To complete the information needed for simulation purposes the following categories of user input must be given:

1. The queueing disciplines followed by entities at each queue.
2. The starting conditions.
3. The system recording functions.

Queueing disciplines are assumed to be first-in, first-out unless otherwise stated. Other disciplines, such as last-in, first-out, random, or maximum of an expression, are readily available. For instance, the next user to start COMPUTE might be the one with the highest priority.

The starting conditions for the simulation are usually chosen to avoid transient conditions associated with starting conditions of "empty and idle." This is easily accomplished by indicating the activities in progress and their completion times. All entities which are not involved in activities in progress are placed in appropriate queues.

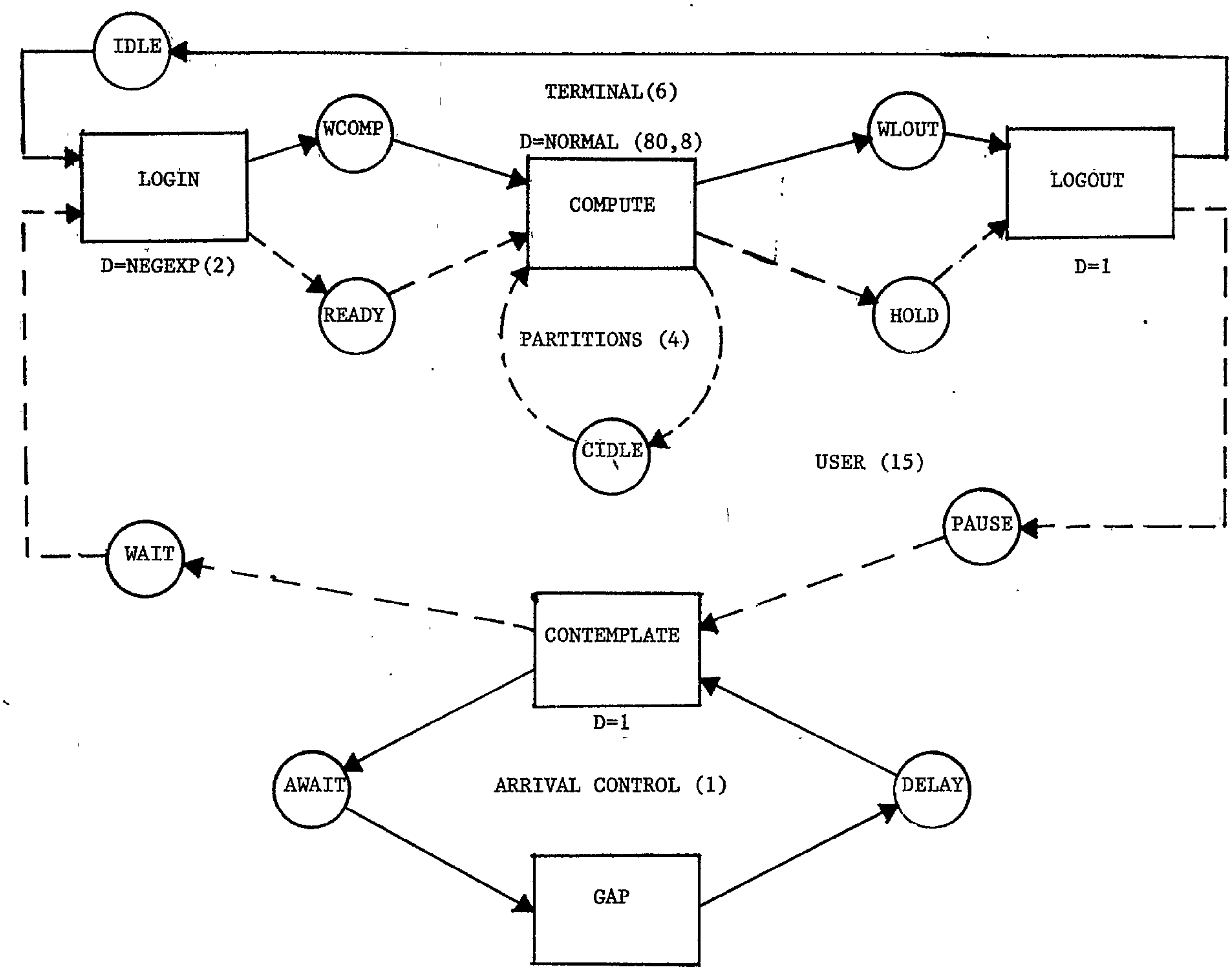
Simulations run with CAPS written programs automatically provide the user with a count for each activity started. In addition, the user can specify the recording of the length and wait time distributions for queues. The final user input required is the length of the simulation.

The example chosen is simple, by design, to illustrate the capabilities of CAPS and its ease of use. The appendix contains a listing of the actual CAPS dialog, with comments; a listing of the code written by CAPS; and the results of executing the code. Finally, the results are given for three executions, modifying MAR, which specifies the arrival rate.

The computing system used was the University of Wisconsin's Univac 1110. The total elapsed time, including 2 user system errors (i.e., not CAPS errors which are corrected), was 46 minutes 29 seconds at a cost of $11.18. The foregoing included the 2 system errors, listing the program on-line, and executing 3 runs of the model. The results for the CAPS dialog and a single execution are 22 minutes 55 seconds and $3.29. The ECSL Code generated consisted of 139 lines. Using $0.002 per

FIGURE 8.

Completed Activity Cycle Diagram



verified character and an average of 25 characters per line, the costs of manually producing the physical code would be $6.95. CAPS costs for both the interaction and producing the code is $0.99, a not insignificant saving. The costs and elapsed times from the interaction given in the Appendix are in Tables 1 and 2. Table 1 follows the history of the interaction, including user system errors. These follow directly the cost history as displayed in the Appendix. In Table 2, the user system errors are eliminated and subtotals provided to highlight the CAPS costs. It is evident that CAPS is cost effective strictly on the basis of producing the simulation code. When the total costs of obtaining simulation results are considered, CAPS should show even greater efficiency and effectiveness when compared with other techniques.

CONCLUSIONS

The ability of the CAPS to provide simulation output in a short time frame has been illustrated for a very simple problem. The use of activity cycles as a basis for system decomposition, for input to CAPS, and for ease of communication of the logical relationships of a system has been demonstrated.

CAPS was demonstrated to be cost effective, considering only key purchasing costs. It is suggested that CAPS's cost effectiveness would be much greater if the total costs of a simulation project were considered.

The CAPS appears to hold promise as a vehicle for making simulation a practical problem solving tool and as a basis for teaching the use of simulation in realistic environments.

TABLE 1.

Elapsed Time and Costs of Appendix Runs

	Time		Cost	
	Δ	Σ	Δ	Σ
Start up	0:00		$0.13	0.13
CAPS Dialog	16:43	16:43	0.99	1.12
List ECSL Code	3:51	20:34	0.24	1.36
User Error	3:14	23:48	2.62	3.98
Execute Model	6:12	30:00	2.30	6.28
User Error	3:40	33:40	0.17	6.45
Execute Model	3:19	36:59	2.01	8.46
Execute Model	9:30	46:29	2.85	11.31

TABLE 2.

Elapsed Time and Costs Excluding User System Errors

CAPS Dialog	16:43		$0.99	
Execute Model	6:12		2.30	
Subtotal		22:55		3.29
Execute Model	3:19		2.01	
Execute Model	9:30		2.85	
Total		35:44		8.15

REFERENCES

1. Clementson, A. T., "Computer Aided Programming for Simulation," University of Birmingham.
2. "Extended Control and Simulation Language - User's Manual," University of Birmingham.
3. Hutchinson, G. K., "An Introduction to Activity Cycles," Simuletter, October, 1975.
4. Hutchinson, G. K., "An Introduction to CAPS," Simuletter, October, 1975.
5. Hills, P. R., "HOCUS - User's Manual," P-E Consulting Group.
6. Clementson, A. T., "CAPS Detailed Reference Manual," University of Birmingham.

APPENDIX

CAPS Interactive Dialog

This Appendix contains the line by line interaction of a CAPS terminal session submitting the activity diagram shown in Figure . In addition to the CAPS dialog the system commands (shown by a leading @) and editing are given. The implementation of CAPS reflects the desire at the University to provide easy access to the system and the availability of utilities of the Univac Exec VIII on the 1110. Implementations at other installations would not necessarily have these functions or formats.

The general outline of the Appendix is: the CAPS dialog, use of the Edit processor to list the CAPS generated ECSL code, execution of the generated program, a user error, and two iterations of editing the the code followed by model execution. The cost accounting (@COST) processor is envoked at each major point in the process to give both cost and elapsed time accounting. These were the basis of the cost analysis in the paper.

The arrival rate in the model is determined by a negative expotential distribution with mean MAR. The three runs vary only in the values of MAR, these being 15, 20, and 12 respectively. Analysis of the output and conclusions about the system design are left to the interested reader.

Unfortunately it is somewhat difficult to differentiate between the originators of individual lines on the listing, i.e., between user input and computer output. Comments have been added that will help. In the most important section, the CAPS dialog, user inputs start 1 character position to the left of computer responses.

```
@COST
DATA IGNORED - IN CONTROL MODE???????@COST
@P
 READY
@COST

FOR Y10704 AT 14:06:55 ON 02-10-77...
SPECIFY ITEM:
TYPE THE NAME OF THE ITEM YOU WISH TO SEE OR 'HELP'
@P
  $.13
 READY
 READY
@ASG,UP COMPUTER.
READY
@H*SS.CAPS  COMPUTER.

 COMPUTER AIDED PROGRAMMING - SIMULATION***C A P S***02/10/77

  UNIVERSITY OF WISCONSIN.CONTACT G.HUTCHINSON 963-4274

 DO YOU WISH TO HAVE INSTRUCTIONAL COMMENTS-
 PARDON - PLEASE ANSWER YES OR NO-
YES
 DURING THIS DISCUSSION YOU WILL BE ASKED FOR A NUMBER OF LISTS.
 WHEN A LIST IS COMPLETE A BLANK LINE SHOULD BE ENTERED.
 IF WHEN TYPING YOU MAKE ERRORS, THESE MAY BE CORRECTED
 BY BACKSPACING (ERACES LAST CHARACTER) USE THE CTRL+H KEYS
  OR ERACE FIELD (ERACES LINE)USE THE CTRL + X KEYS
 WHEN A LINE IS COMPLETE PRESS  EOF.
 PLEASE NOTE-ONCE EOF HAS BEEN PRESSED
 IT MAY NOT BE POSSIBLE TO MAKE CORRECTIONS IMMEDIATELY
  THE DISCUSSION IS IN FIVE SECTIONS
 LOGIC-PRIORITIES-ARITHMETIC-RECORDING-INITIAL CONDITIONS
 AT THE END OF EACH SECTION IT IS PPOSSIBLE
 TO RETURN TO THE BEGINNING OF ANY EARLIER SECTION

 DO YOU WISH TO START A NEW PROBLEM
YES
 PROBLEM NAME -
USERSIM

  TYPE NAME OF ENTITIES
USER
 HOW MANY
15
 TYPE A LIST OF THE STATES THROUGH WHICH THESE ENTITIES PASS.
 THIS SHOULD CONSIST OF EITHER-
    A) AN ALTERNATION OF QUEUES AND ACTIVITIES
       STARTING AND ENDING WITH A QUEUE
 OR B) A LIST OF ACTIVITIES ALONE
 PRECEDE QUEUES BY Q AND ACTIVITIES BY A
QWAIT
ALOGIN
QREADY
ACOMPUTE
QHOLD
ALOGOUT
QPAUSE
ACONTEMPLATE
```

```
QQWAIT

 EVERY CYCLE MUST BE CLOSED
 THUS THE FIRST QUEUE MUST REOCCUR AND
 THE LAST QUEUE MUST HAVE OCCURED BEFORE
 CYCLE NOT COMPLETE - DO YOU WISH TO CONTINUE IT -
NO
 USER  CYCLE DELETED

  TYPE NAME OF ENTITIES
USER
 HOW MANY
15
 TYPE LIST OF STATES AS ABOVE
QWAIT
ALOGIN
QREADY
ACOMPUTE
QHOLD
ALOGOUT
QPAUSE
ACONTEMPLATE
QWAIT

 IS THIS CYCLE CORRECT
 PARDON - PLEASE ANSWER YES OR NO -
YES

  TYPE NAME OF ENTITIES
TERMINAL
 HOW MANY
6
 TYPE LIST OF STATES AS ABOVE
QIDLE
ALOGIN
QWCOMP
ACOMPUTE
QWLOUT
ALOGOUT
QIDLE

 IS THIS CUCLE CORRECT
YES

  TYPE NAME OF ENTITIES
COMPUTER
 YOU HAVE ALREADY USED THAT WORD IN A DIFFERENT WAY

  TYPE NAME OF ENTITIES
CPU
 HOW MANY
4
 TYPE LIST OF STATES AS ABOVE
QCIDLE
ACOMPUTE
QIDLE
 NAME IDLE  REUSED ILLEGALLY
 A QUEUE CANNOT BE IN TWO DIFFERENT CYCLES
QCIDLE

 IS THIS CYCLE CORRECT
YES
 HOW MANY
1
 TYPE LIST OF STATES AS ABOVE
QAWAIT
AGAP
QDELAY
ACONTEMPLATE
QAWAIT
```

```
 IS THIS CYCLE CORRECT
YES

  TYPE NAME OF ENTITIES

 ARE THERE ANY ACTIVITIES WHICH USE MORE THAN ONE
 ENTITY OF A PARTICULAR TYPE-
NO

 FROM WHAT YOU SAID SO FAR, THE FOLLOWING ARE THE
 MAXIMUM NUMBER OF SIMULTANEOUS REALISATION OF THE ACTIVITIES
 ACTIVITY NUMBER
 LOGIN    6
 COMPUTE  4
 LOGOUT   6
 CONTEM   1
 GAP,     1
 DO YOU WISH TO APPLY ANY LOWER LIMITS
NO
 ACTIVITY LOGOUT APPEARS TO BE BOUND TO COMPUT
 I.E. THE FOLLOWING QUEUES ARE DUMMIES
 HOLD
 WLOUT
 DO YOU AGREE
YES
 DO YOU WISH TO SEE A SUMMARY OF THE CYCLES
YES
 USER     15 QWAIT   ALOGIN  QREADY  ACOMPUT Q       ALOGOUT QPAUSE
             ACONTEM QWAIT
 TERMIN   6  QIDLE   ALOGIN  QWCOMP  ACOMPUT Q       ALOGOUT QIDLE
 CPU      4  QCIDLE  ACOMPUT QCIDLE
 ARRIVA   1  QAWAIT  AGAP    QDELAY  ACONTEM QAWAIT

 LOGIN  USES 1 USER   1 TERMIN
 COMPUT USES 1 USER   1 TERMIN 1 CPU
 LOGOUT USES 1 USER   1 TERMIN
 CONTEM USES 1 USER   1 ARRIVA
 GAP    USES 1 ARRIVA
  DO YOU WISH TO MAKE ANY CHANGES IN THE LOGIC SECTION
NO

 PRIORITIES
 ARE THERE ANY QUEUES WHOSE DISCIPLINE IS NOT F-I-F-O -
NO

  THE FOLLOWING ARE BOUND ACTIVITIES
 (A BOUND ACTIVITY IS ONE WHICH WILL ALWAYS START IMMEDIATELY
 UPON THE COMPLETION OF THE PRECEDING ACTIVITY)
 LOGOUT

  THE ORDER OF THE FOLLOWING ACTIVITIES IS UNIMPORTANT
 CONTEM
 GAP
 LOGIN
  DO YOU WISH TO MAKE ANY CHANGES IN THE PRIORITY SECTION
NO

 ARITHMETIC
 AFTER EACH ACTIVITY NAME, TYPE FORMULA FOR ITS DURATION
 CONTEM=
1
 COMPUT=
NORMAL(80 , 8, ABC)
 GAP    =
NEGEXP(MAR, RA)
 LOGIN =
NEGEXP(2,RB)
```

```
 LOGOUT=
1

 IN WHICH ACTIVITY IS ABC    EVALUATED-
 (N.B. IF VARIABLE IS NOT TO BE EVALUATED BY ANY ACTIVITY,
 JUST TYPE EOF)

 WHAT IS ITS INITIAL VALUE-
1241
 IN WHICH ACTIVITY IS MAR    EVALUATED-

 WHAT IS ITS INITIAL VALUE-
15
 IN WHICH ACTIVITY IS RA     EVALUATED-

 WHAT IS ITS INITIAL VALUE-
1153
 IN WHICH ACTIVITY IS RB     EVALUATED-

 WHAT IS ITS INITIAL VALUE-
441
 DO YOU WISH TO DEFINE ANY OTHER ATTRIBUTES FOR ENTITIES
NO
  DO YOU WISH TO MAKE ANY CHANGES IN THE ARITHMETIC SECTION
NO
 RECORDING
 TWO KINDS OF RECORDING MAY BE INCLUDED
 1)LENGTH OF QUEUE
 2)LENGTH OF TIME ENTITY IS DELAYED IN QUEUE
 TYPE, AFTER THE QUEUE NAME, WHICH KIND OF RECORDING IS REQUIRED
 TYPE 0, IF NO RECORDING REQUIRED
 TYPE 3, IF BOTH KINDS ARE REQUIRED
 READY =
3
 PAUSE =
0
 WAIT  =
3
 WCOMP =
0
 IDLE  =
3
 CIDLE =
3
 DELAY =
0
 AWAIT =
0
 FOR EACH QUEUE FOR WHICH DELAYS ARE TO BE RECORDED

100
 WAIT    RANGE=0 TO
50
 IDLE    RANGE=0 TO
30
 CIDLE   RANGE=0 TO
30
  DO YOU WISH TO MAKE ANY CHANGES IN THE RECORDING SECTION
NO

  INITIAL CONDITION
 ARE THERE ANY ACTIVITIES IN PROGRESS
YES
 (NOTE-TERMINATION TIMES MUST BE CONSTANTS)
 ACTIVITY -
LOGIN
 TERMINATION TIME =
1
 TERMINATION TIME =
3
```

```
 TERMINATION TIME =

 ACTIVITY -
COMPUTE
 TERMINATION TIME =
22
 TERMINATION TIME =
43
 TERMINATION TIME =
55
 TERMINATION TIME =

 ACTIVITY -

  TYPE HOW MANY ENTITIES SHOULD BE IN EACH QUEUE LISTED
 AFTER THE QUEUE NAME
 USER   - 15 ENTITIES
  5 USED BY ACTIVITIES IN PROGRESS
 READY  -
0
 PAUSE  -
14
 ONLY 10 LEFT - TRY AGAIN
10
 TERMIN - 6 ENTITIES
  5 USED BY ACTIVITIES IN PROGRESS
 WCOMP  -
0
 IDLE   -
1
 CPU    - 4 ENTITIES
  3 USED BY ACTIVITIES IN PROGRESS
 CIDLE  -
1
 ARRIVA - 1 ENTITIES
 DELAY  -
0
 AWAIT  -
1
 PLEASE GIVE THE DURATION OF THE SIMULATION
1000
  DO YOU WISH TO MAKE ANY CHANGES IN THE INITIAL CONDITION SECTION
NO

YES

 YOUR CAPS GENERATED PROGRAM,IN ECSL IS IN FILE COMPUTER.

 DO YOU WISH TO SEE ADDITIONAL OPTIONS?
YES

  YOU MAY WISH TO PRINT,EDIT, OR SAVE YOUR FILE:
 TO PRINT:  @EDIT,U (FILE)
            P 500
 TO EDIT- SEE EDIT MANUAL
 TO SAVE:   @SAVE,S (FILE),(DATE)
 --------------------------------
 TO COMPILE ONLINE: @ADD,P H*SS.RUNOL
                    @ADD (FILE)
  N.B. ONLINE COMPUTING IS RELATIVELY EXPENSIVE!
 TO RUN BATCH ADD TO FRONT OF OF FILE:
  @RUN (PARAMETERS)
  @ADD H*SS.RUN
     OUTPUT IS HIGH SPEED PRINTER
  USE @SYM FOR REMOTE OUTPUT(SEE USER-S MANUAL)

  CAPS AND MACC BID YOU ADIEU.
```

```
@COST
DATA IGNORED - IN CONTROL MODE???@COST
@P
 READY
@COST
FOR Y10704 AT 14:23:38 ON 02-10-77...
SPECIFY ITEM:
@P
 $1.12
 READY
 READY
@EDIT,U COMPUTER.
EDIT 1.39-2/10-14:24
EDIT
:P  555
*      COMPILE USER
       THERE ARE 15 USER   SET  READY  PAUSE  WAIT    WITH  TIME
       THERE ARE  6 TERMIN SET  WCOMP  IDLE      WITH  TIME
       THERE ARE  4 CPU    SET  CIDLE    WITH   TIME
       THERE ARE  1 ARRIVA SET  DELAY  AWAIT
       FUNCTION PICTURE NEGEXP  NORMAL
       HIST ZAREADY (USER  0,1)
       HIST WREADY (10,  5,  10)
       HIST ZBWAIT  (USER  0,1)
       HIST WWAIT  (10,  2,   5)
       HIST ZCIDLE  (TERMIN 0,1)
       HIST WIDLE  (10,  1,   3)
       HIST ZDCIDLE (CPU    0,1)
       HIST WCIDLE (10,  1,   3)
       DURATION=     1
       CHAIN
         USER   1  INTO READY  AFTER  DURATION
         TIME OF USER   1 = DURATION
         TERMIN 1  INTO WCOMP  AFTER  DURATION
       DURATION=     3
       CHAIN
         USER   2  INTO READY  AFTER  DURATION
         TERMIN 2  INTO WCOMP  AFTER  DURATION
       DURATION=    22
       ADURATION= DURATION+1
       CHAIN
         USER   3  INTO PAUSE  AFTER ADURATION
         TERMIN 3  INTO IDLE   AFTER ADURATION
         TIME OF TERMIN 3 =ADURATION
         CPU    1  INTO CIDLE  AFTER  DURATION
       DURATION=    43
       ADURATION= DURATION+1
       CHAIN
         USER   4  INTO PAUSE  AFTER ADURATION
         TERMIN 4  INTO IDLE   AFTER ADURATION
         TIME OF TERMIN 4 =ADURATION
         CPU    2  INTO CIDLE  AFTER  DURATION
         TIME OF CPU    2 = DURATION
       DURATION=    55
       ADURATION= DURATION+1
       CHAIN
         USER   5  INTO PAUSE  AFTER ADURATION
         TERMIN 5  INTO IDLE   AFTER ADURATION
         TIME OF TERMIN 5 =ADURATION
         CPU    3  INTO CIDLE  AFTER .DURATION
         TIME OF CPU    3 = DURATION
       RECYCLE
       ACTIVITIES  1000
       BEGIN RECORD
       DURATION=CLOCK-PREVCLOCK
       PREVCLOCK=CLOCK
       ADD A TO ZAREADY , DURATION
       ADD B TO ZBWAIT  , DURATION
       ADD C TO ZCIDLE  , DURATION
       ADD D TO ZDCIDLE , DURATION
```

```
BEGIN CONTEM
FIND   FIRST USER    A  IN PAUSE
FIND   FIRST ARRIVA B  IN DELAY
DURATION=1
CONTEM+1
CHAIN
  USER   A  FROM PAUSE  INTO WAIT   AFTER  DURATION
  TIME OF USER    A = DURATION
  ARRIVA B  FROM DELAY  INTO AWAIT  AFTER  DURATION
REPEAT
BEGIN COMPUT
FIND   FIRST USER    A  IN READY
FIND   FIRST TERMIN B  IN WCOMP
FIND   FIRST CPU     C  IN CIDLE
DURATION=NORMAL( 80, 8 , ABC    )
COMPUT+1
ADURATION= DURATION+1
CHAIN
  USER    A  FROM READY  INTO PAUSE  AFTER ADURATION
  ADD -TIME OF USER    A  TO WREADY
  TERMIN B  FROM WCOMP  INTO  IDLE  AFTER ADURATION
  TIME OF TERMIN B = ADURATION
  CPU     C  FROM CIDLE  INTO CIDLE  AFTER  DURATION
  ADD -TIME OF CPU     C  TO WCIDLE
  TIME OF CPU     C = DURATION
REPEAT
BEGAN GAP
FIND   FIRST ARRIVA A  IN AWAIT
DURATION=NEGEXP( MAR    , RA     )
GAP   +1
  ARRIVA A  FROM AWAIT  INTO DELAY  AFTER  DURATION
REPEAT
BEGIN LOGIN
FIND   FIRST USER    A  IN WAIT
FIND   FIRST TERMIN B  IN IDLE
DURATION=NEGEXP( 2 , RB     )
LOGIN+1
CHAIN
  USER    A  FROM WAIT   INTO READY  AFTER  DURATION
  ADD -TIME OF USER    A  TO WWAIT
  TIME OF USER    A = DURATION
  TERMIN B  FROM IDLE   INTO WCOMP  AFTER  DURATION
  ADD -TIME OF TERMIN B  TO WIDLE
REPEAT
BEGIN COUNT QUEUES
COUNT A IN READY
COUNT B IN WAIT
COUNT C IN IDLE
COUNT D IN CIDLE
FINALISATION
PRINT'CONTEM WAS STARTED'CONTEM' TIMES'
PRINT'COMPUT WAS STARTED'COMPUT' TIMES'
PRINT'GAP    WAS STARTED'GAP    ' TIMES'
PRINT'LOGIN  WAS STARTED'LOGIN ' TIMES'
PRINT/'HISTOGRAM OF LENGTH OF QUEUE READY '
PICTURE(ZAREADY)
PRINT/'HISTOGRAM OF DELAYS AT READY'
PICTURE(WREADY)
PRINT/'HISTOGRAM OF LENGTH OF QUEUE WAIT'
PICTURE(ZBWAIT)
PRINT/'HISTOGRAM OF DELAYS AT WAIT'
PICTURE(WWAIT)
PRINT/'HISTOGRAM OF LENGTH OF QUEUE IDLE'
PICTURE(ZCIDLE)
PRINT/'HISTOGRAM OF DELAYS AT IDLE'
PICTURE(WIDLE)
PRINT/'HISTOGRAM OF LENGTH OF QUEUE CIDLE'
PICTURE(ZDCIDLE)
```

```
      PRINT/'HISTOGRAM OF DELAYS AT CIDLE'
      PICTURE(WCIDLE)
      DATA
PAUSE   6 TO 15
IDLE       6
CIDLE      4
AWAIT      1
RB       441
RA      1153
MAR       15
ABC     1241
      END
*     EXECUTE
*** TOP OF FILE ***
:@P
NOTHING CHANGED, NOTHING FILED
READY
@XQT H*SS.ECSL
E.C.S.L. SYSTEM - UNIVERSITY OF WISCONSIN
@ADD
@ADD COMPUTER
*     COMPILE USER           NOLIST  NOTABLE

                 LIST DENSITY 36496 PER CENT

E.C.S.L. SYSTEM - UNIVERSITY OF WISCONSIN

*     EXECUTE

E.C.S.L. SYSTEM                     PROGRAM - USERNOLI
CUTED ON  2/10/77                     PAGE   1
CONTEM WAS STARTED           59 TIMES
COMPUT WAS STARTED           50 TIMES
GAP    WAS STARTED           60 TIMES
LOGIN  WAS STARTED           50 TIMES

HISTOGRAM OF LENGTH OF QUEUE READY
  CELL  FREQUENCY
     0    195***************************************
     1    347*********************************************************************
     2    458*******************************************************************************************

HISTOGRAM OF DELAYS AT READY
  CELL  FREQUENCY
     5     10**********
    15     13*************
    25     11***********
    35      9*********
    45      2**
    55      5*****

HISTOGRAM OF LENGTH OF QUEUE WAIT
  CELL  FREQUENCY
     0    625***********************************************************************************************
********
     1    135*******************
     2     46******
     3     29****
     4     29****
     5     23***
     6      5
     7     45******
     8     25***
     9     38*****
```

```
HISTOGRAM OF DELAYS AT WAIT
  CELL  FREQUENCY
     2    31*******************************
     7     3***
    12     0
    17     4****
    22     0
    27     2**
    32     2**
    37     0
    42     0
    47     8********

HISTOGRAM OF LENGTH OF QUEUE IDLE
  CELL  FREQUENCY
     0   529****************************************************************************************
     1   306***************************************************
     2   134**********************
     3    24****
     4     7*

HISTOGRAM OF DELAYS AT IDLE
  CELL  FREQUENCY

E.C.S.L. SYSTEM  UNIVERSITY OF WISCONSIN      PROGRAM - USERNOLI
   EXECUTED ON  2/10/77                             PAGE    2

     1    24************************
     4     2**
     7     2**
    10     3***
    13     1*
    16     1*
    19     0
    22     3***
    25     3***
    28    11***********

HISTOGRAM OF LENGTH OF QUEUE CIDLE
  CELL  FREQUENCY
     0   963******************************************************************************************
******
     1    27**
     2    10*

HISTOGRAM OF DELAYS AT CIDLE
  CELL  FREQUENCY
     1    48************************************************
     4     0
     7     0
    10     0
    13     0
    16     0
    19     1*
    22     0
    25     1*

E.C.S.L. SYSTEM - UNIVERSITY OF WISCONSIN

 COST
FOR Y10704 AT 14:36:55 ON 02-10-77...
SPECIFY ITEM:
@P
 $6.28
 READY
 READY
```