UAH Research Report No. 725

Automatic Programming of Simulation Models

Final Report
Task III

Bernard J. Schroer
Fan T. Tseng
Shou X. Zhang
Wen S. Dwan
Johnson Research Center
University of Alabama in Huntsville
Huntsville, Alabama 35899
(205) 895-6361

NASA/MSFC Grant NAG8-641

September 1988

ABSTRACT

This report presents the results for Task III, Automatic Programming of Simulation Models, on NASA/MSFC Grant NAG8-641. Mr. John W. Wolfsberger was the Technical Monitor. The research was also funded in part by contract ADECA-UAH-9001 from the Science, Technology, and Energy Division, Alabama Department of Economic and Community Affairs.

TABLE OF CONTENTS

1.0 INTRODUCTION

The term Automatic Programming (AP) has been defined as an application of Artificial Intelligence (AI) dealing with automating some aspects of the computer programming process. In other words, automatic programming uses another program, an AP system, to raise the level of specifying program instructions. Stated from the practitioner's point of view, AP systems are programs that help programmers write programs.

The objective of automatic programming is to improve the overall environment for describing the program. This improved environment is realized by a reduction in the amount of detail that the programmer needs to know and is exposed to. Furthermore, this improved environment is achieved by a specification language that is more natural to the user's problem domain and to the user's way of thinking and looking at the problem.

The goal of this research is to apply the concepts of automatic programming (AP) to modeling discrete event simulation systems. Specific emphasis is on the design and development of simulation tools to assist the modeler define or construct a model of the system and to then automatically write the corresponding simulation code in the target simulation language, GPSS/PC. A related goal is to evaluate the feasibility of various languages for constructing automatic programming simulation tools.

The following domains have been selected for the construction and evaluation of these AP simulation tools:

- Manufacturing systems that can be described with assembly and subassembly lines where parts are being added to an assembly; manufacturing cells that provide parts to the assembly and subassembly lines; and inventory transfers of parts between the

manufacturing cells and assembly lines.

- Reliability networks that can be described in terms of activities with starting and ending nodes; activity times; activity failures and repairs; and the effect or the interdependency of failures on other activities.

2.0 TRADITIONAL SIMULATION APPROACH

Figure 1 outlines the traditional approach to simulation modeling. The traditional approach requires that the simulationist first have a thorough understanding of the problem or system to be modeled. Given this knowledge, the simulationist must then define the model of the system resulting in a detailed problem specification.

The next step is for the simulationist to write the simulation program, following the problem specification, in the selected target language. The simulation program is then debugged to remove all syntax errors. Following debugging, the model must be verified and validated. Next, the user defines the experiment and the model is executed.

3.0 ARTIFICIAL INTELLIGENCE AND SIMULATION

The current research into coupling AI with the traditional simulation approach is concentrated into two specific areas. The first area is the use of AI concepts for automating the problem specification process. The second area, and the more difficult area, is the use of AI for automating the writing of the executable code in the target simulation language.

3.1 Automatic Problem Specification

Figure 2 is a schematic of the automatic problem specification approach for coupling AI with simulation. This approach can be considered

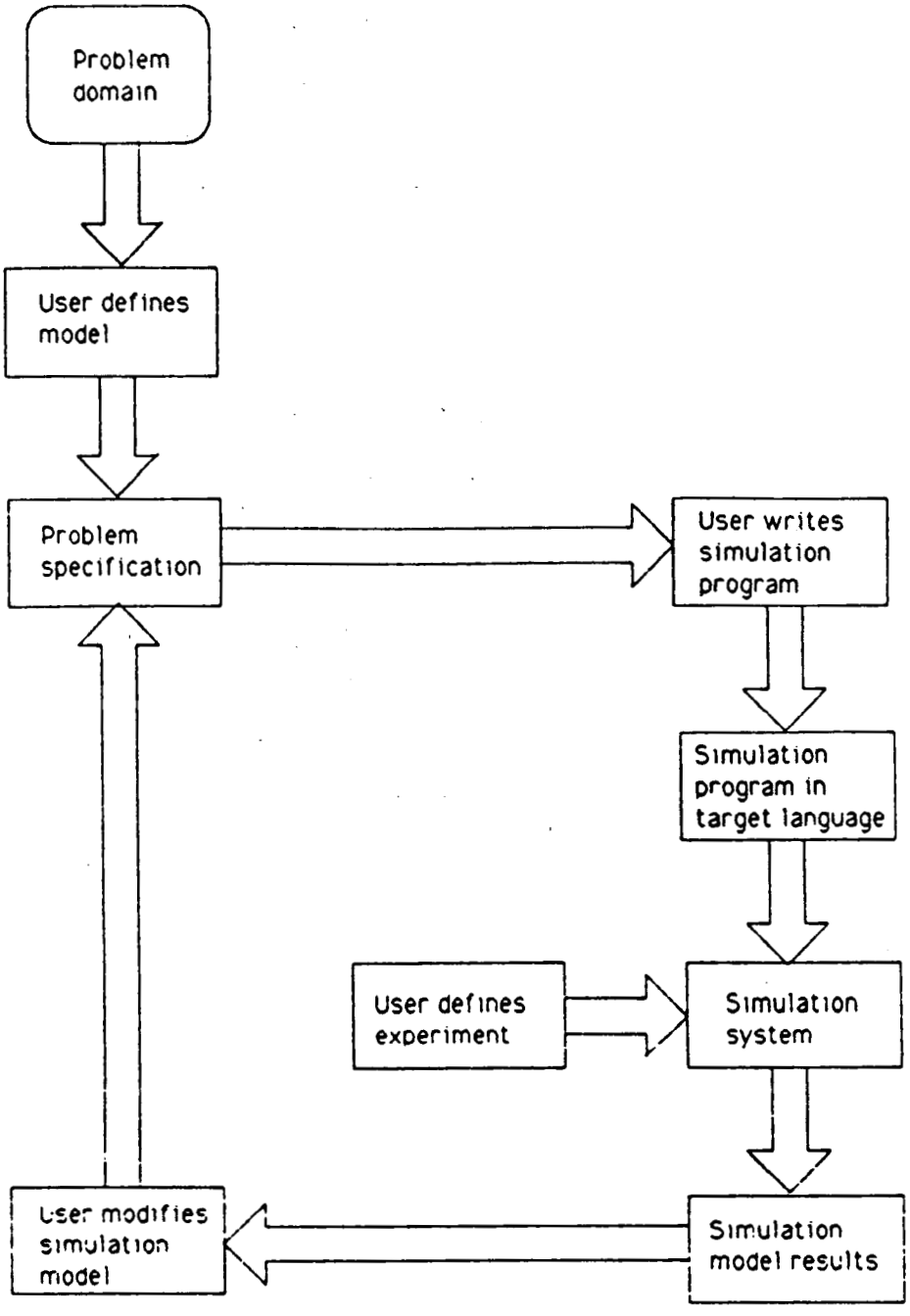

Figure 1. Traditional simulation approach

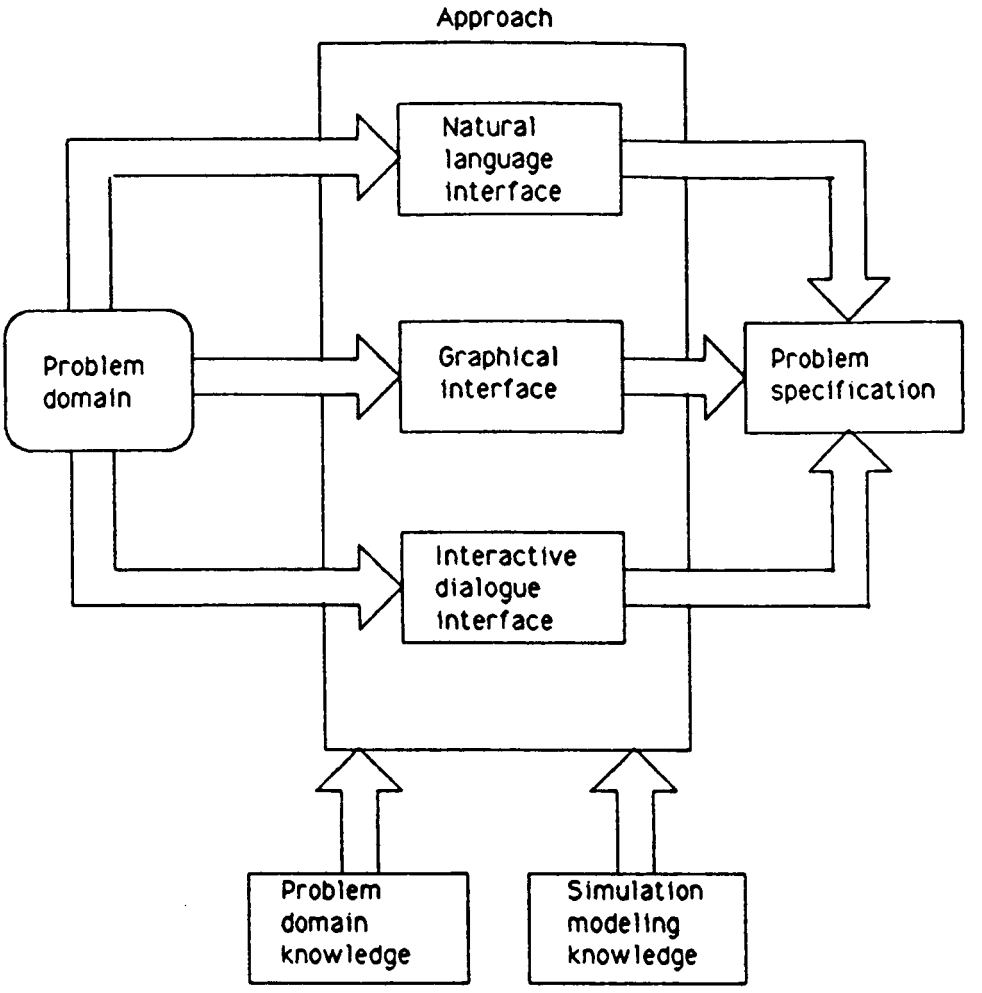

Figure 2. Automatic problem specification

as an intelligent assistant to the user in defining the simulation model. Some authors call this approach the specification acquisition element of the simulation model construction (Murray and Sheppard 1988).

Two types of knowledge, or knowledge bases, are used to support the automatic problem specification. These types of knowledge are problem domain knowledge and simulation modeling knowledge. The domain knowledge is knowledge related to a specific problem. Consequently, each problem domain requires its own knowledge base. For example, the Electronics Manufacturing Simulation System (Ford and Schroer 1987) is domain specific to electronics assembly. The simulation modeling knowledge is knowledge necessary in constructing the final simulation code in the target language.

These knowledge bases are then used in defining a set of rules that govern the interactive dialogue between the user and the system. The system will execute different logic by firing different rules based on the user's response to specific questions. At the completion of this interactive dialogue, the system will have defined an internal specification of the problem or model.

Three approaches are commonly used to assist the user define the simulation model, or problem specification. These approaches are a natural language interface, an interactive graphical interface, and an interactive dialogue interface.

The natural language interface (NLI) lets the user specify the problem in free text format. The NLI then attempts to parse the text and automatically generate the simulation code in the target language. One of the earliest NLI's was the natural language programming for queueing simulations (NLPQ) (Heidorn 1974). Through an interactive dialogue in English, the NLPQ system creates an internal description of the queueing

problem and generates the simulation code in the target language GPSS. A more recent NLI was the Electronic Manufacturing Simulation System (EMSS) (Ford and Schroer 1987). The system is constructed around a dictionary of electronics assembly words and expressions. The EMSS parses the text into a conceptual dependency representation which is used to automatically write the simulation code in the target language SIMAN.

The second approach to assist the user specify the problem, which is less difficult than the NLI, is an interactive graphical interface (IGI). An IGI consists of a menu of icons that are mouse selectable, to construct a graphical representation of the system. Once the system has been constructed, the user inputs the attributes corresponding to the icons. Khoshnevis and Chen (1986) have developed an object oriented approach for graphically modeling a system. A library of icons is available to the user in constructing the model. The system is rule based and written in common LISP on an IBM PC. Once the graphical description of the model is completed, the system automatically generates the equivalent SLAM simulation code.

The third approach to assist the user define the problem specification is the interactive dialogue interface. This approach is similar to the graphical interface with the exception of the icons. Instead, the user responds to a series of questions in defining the problem specification. Several systems have been developed using the interactive dialogue approach. Haddock and Davis (1985) have developed a flexible manufacturing system (FMS) simulation generator. Brazier and Shannon (1987) have developed an automatic programming system for modeling automated guided vehicle systems. The system is written in Turbo Prolog for an IBM PC and generates SIMAN code. More recently, a knowledge based model construction system has

been developed to automate model definition and code generation (Murray and Sheppard 1988). The system automatically writes executable SIMAN code.

3.2 Automatic Simulation Writing

Figure 3 is a schematic of the automatic code generation approach for coupling AI with simulation. Basically, two approaches exist for taking the internal problem specification and then automatically generating the executable code in the target simulation language. The first approach is to generate the simulation code directly from the internal representation of the problem specification.

The second approach is to use a library of predefined macros to assist in the automatic generation of the simulation code. The advantage of such an approach is the ability to solve more complex problems than those previously discussed in the literature. The disadvantage is that most macros are domain specific. As a result, additional macros are needed to solve another problem domain.

Two types of knowledge are used to support the automatic simulation writing. These types of knowledge are simulation modeling knowledge and target language knowledge. The simulation modeling knowledge is the same knowledge base used for the automatic problem specification in Figure 2. The target language knowledge is knowledge specific to a simulation language. These knowledge bases are then used in defining a set of rules for constructing the appropriate lines of simulation code from the internal problem specification.

4.0 SCOPE OF RESEARCH

The following automatic programming simulation tools have been completed during the first year of the contract:

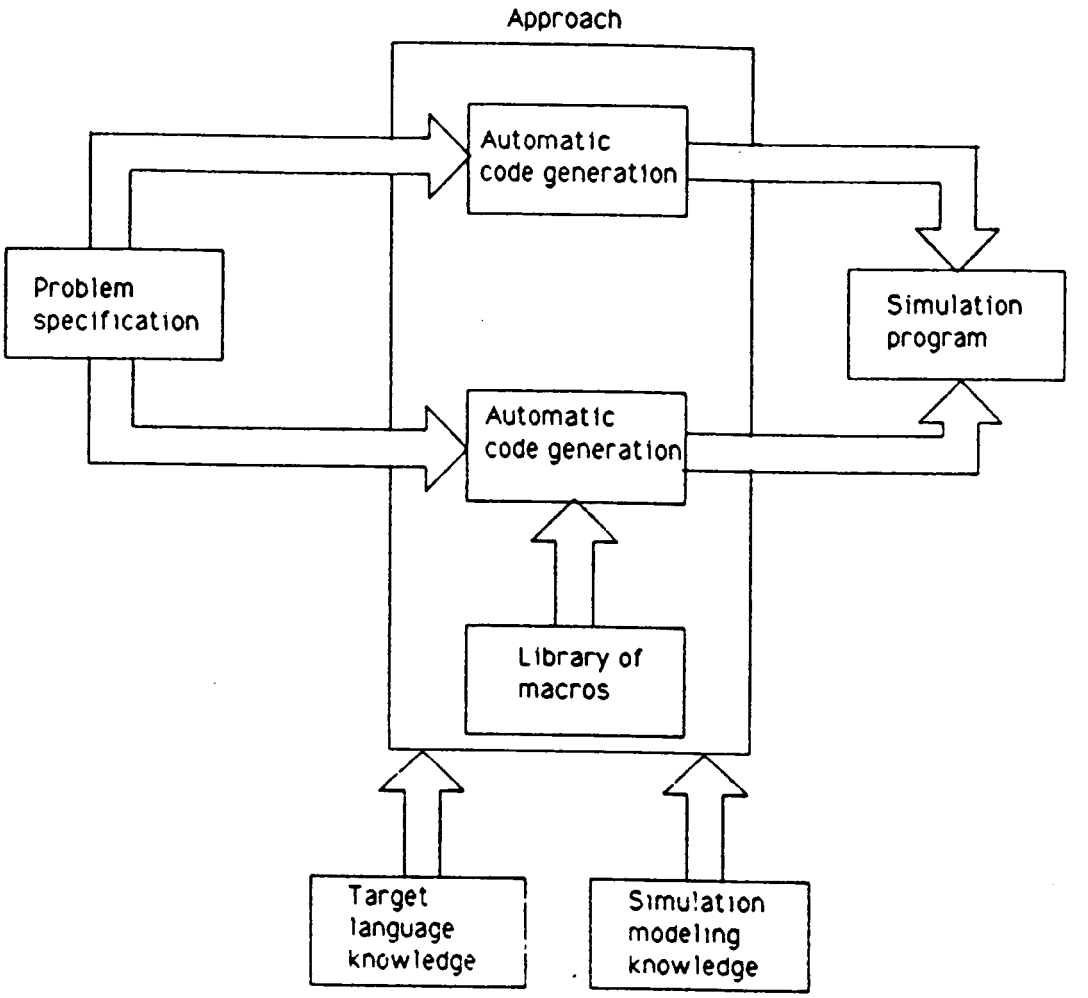

Figure 3. Automatic simulation writing

- AMPS - Automated Manufacturing Programming System for the Symbolics 3620 Workstation
- ANPS/PC - Automated Network Programming System for the PC

Also the following automatic programming simulation tools are under construction and evaluation and should be completed during the second year:

- AMPS/Graphics - Automatic Manufacturing Programming System (using a graphical interface) for the Symbolics 3620 Workstation
- AMPS/PC - A PC version of AMPS written in Turbo Pascal
- ANPS/MAC II - An Apple MAC II version of ANPS written in HyperCard

Table I gives a summary of the simulation tool development and the development platforms. The following two sections will briefly discuss the AP systems that were completed during this contract period. These two AP systems are the AMPS for the Symbolics 3620 and the ANPS for the PC.

5.0 AUTOMATIC MANUFACTURING PROGRAMMING SYSTEM

The Automatic Manufacturing Programming System (AMPS) is a prototype AI based system that has been structured using the conceptual foundation outlined in the previous sections. Specifically, the AMPS system is a simulation assistant to assist the modeler of manufacturing systems define his problem through an interactive user dialogue and to then automatically generate the corresponding GPSS/PC simulation code.

The AMPS system domain is those manufacturing systems that can be described as having:

- Assembly and subassembly lines where parts are being added to an assembly.
- Manufacturing cells that are providing parts to the assembly and

Table I. Development Platforms

Systems	Hardware	Software	Status	Documentation
AMPS	Symbolics 3620	LISP	Complete	UAH Report #659 UAH Report #720
AMPS	Texas Instruments Explorer	LISP	Complete	
ANPS/PC	PC	Turbo Prolog	Complete	UAH Report #704
AMPS/Graphics	Symbolics 3620	Lisp windows and flavors	Under development	
AMPS/PC	PC	Turbo Pascal	Under development	
ANPS/MAC	MAC II	HyperCard	Under development	
AMPS/PC	PC	Turbo C	Not started	

subassembly lines.

- Inventory of parts being moved between the manufacturing cells and subassembly lines.

5.1 AMPS System Overview

Figure 4 is an overview of the AMPS system operation. Once the user has scoped the problem domain, the user sits at the Symbolics 3620 AI workstation and responds to the questions from the interface program. Based on the responses, the interface program creates an internal problem specification file. This file includes the manufacturing process network flow and the attributes for all stations, cells and stock points. The problem specification file is then used as input to the automatic code generator program which generates the simulation program in the target language GPSS/PC. The output of the code generator program is a GPSS/PC program file which is then downloaded to an IBM PC.

The GPSS/PC system (Minuteman 1986) is resident on the PC. The user then adds the experimental frame, such as the run statements, and the GPSS/PC simulation program is executed. The output file is stored on a diskette or printed on the PC. To change the GPSS/PC model, the user returns to the Symbolics 3620 and recalls the problem specification. The user interface then provides the simulationist with a number of options to change or modify the problem specification. The code generator will then rewrite the GPSS program.

The AMPS system is written in LISP for the Symbolics 3620 workstation. The library of macros is written in GPSS/PC. AMPS contains 750 lines of LISP code. The simulation code generated by AMPS is GPSS/PC (Minuteman 1986).

5.2 Library of GPSS Macros

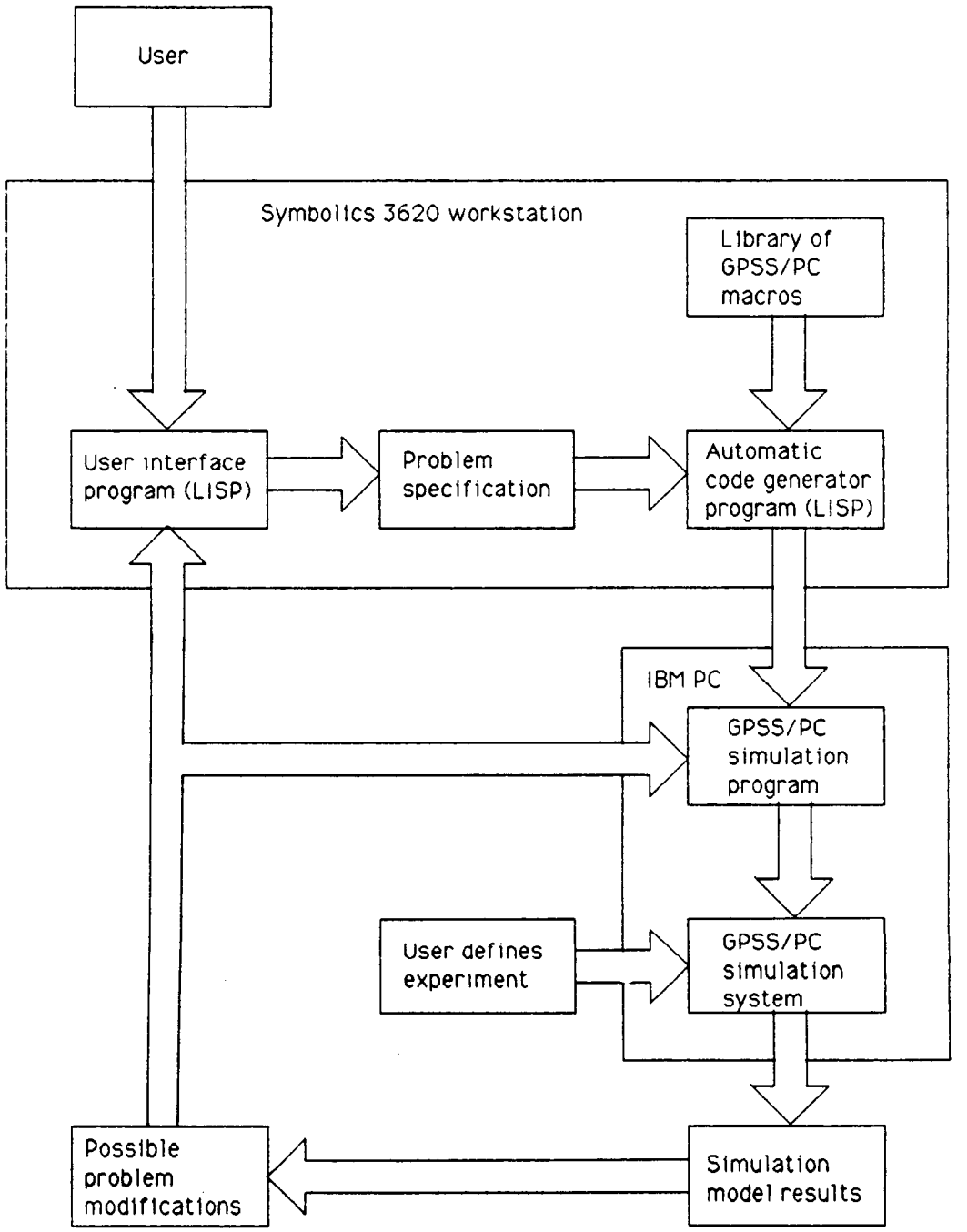

Figure 4. AMPS system overview

In analyzing most manufacturing systems at the macro level, the following functions are generally similar in nature:

- Assembly - adding part X to part Y resulting in part Z
- Fabrication - making of part X from part Y
- Inspection - inspecting part X
- Inventory transfer - moving part X or a cart of part X from stock point A to stock point B
- Simple operation - performing an operation on part X resulting in a modified part X

These five functions represent the current domain of manufacturing functions within the AMPS system. Once the manufacturing functions have been defined, the GPSS subroutines are written for the functions. These routines constitute a library of predefined GPSS subroutines or macros. This library of macros is then called, when needed, in the construction of the GPSS simulation model. Currently, the AMPS system has the following five GPSS subroutines:

- Assembly station
- Manufacturing cell
- Inventory transfer
- Inspection station
- Task station.

Figure 5 briefly describes each of these macros. For example, the assembly station macro has the capability of simulating the adding of a variety of different items to the incoming part resulting in a modified part that is then transferred to the next destination, a station or stock point. For example, in Figure 5, station STA1 assembles two part C's and three part D's to the incoming part A resulting in Part B.

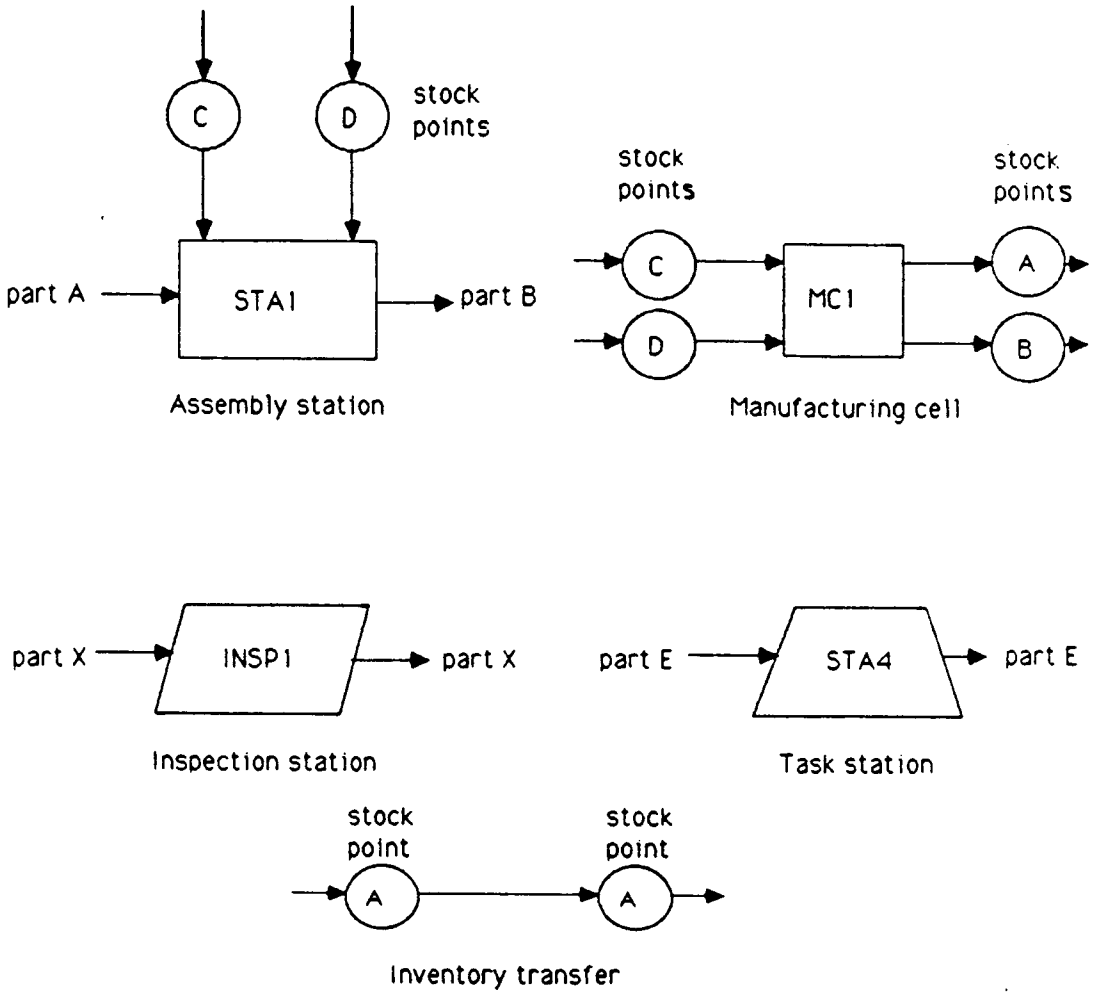

Figure 5. GPSS manufacturing macros

The manufacturing cell makes a cart of specified parts when an order is received. The cell can make multiple part types. For example, cell MC1 makes one part A from two part C's and three part D's and one part B from one part D. The task station performs an operation on a part. For example, in Figure 5 an operation is performed at station STA4 on part E resulting in a modified part E. The inspection station inspects a defined percentage of parts. Of those inspected, a defined percentage is defective. Of those defective, a defined percentage is scrapped.

The inventory transfer macro grants part requests from an assembly station or a manufacturing cell and checks if the inventory system is a push or pull. For a pull system the macro orders a cart of parts by sending an empty cart back to the source and sends a full cart of parts to the demand stock point from the source stock point.

Figure 6 is the GPSS code for the manufacturing cell macro. Note that the matrix savevalues are used to pass data from the main program or from a subroutine to a subroutine.

5.3 Sample Problem

Figure 7 is an example of a typical manufacturing system that can be modeled by the AMPS system. The manufacturing system consists of one assembly line, two subassembly lines, and two manufacturing cells. The assembly line consists of two assembly stations, one task station and one inspection station. Subassembly line 2 consists of one assembly station and one task station while line 3 consists of two assembly stations. Manufacturing cell MC1 provides part type C for assembly station ASSY1 and part type H for assembly station ASSY8. Manufacturing cell MC2 provides part type E for assembly station ASSY5 and part types F and G for assembly station ASSY7. There are a variety of stock points, labeled A through L,

```
4280 ****************************************
4290 *          MANUFACTURING CELL          *
4300 ****************************************
4310 MFG     ASSIGN      13,MX$CELL(P12,1)
4320         ASSIGN      14,MX$CTIME(P12,1)
4330         ASSIGN      16,MX$CTIME(P12,2)
4340         QUEUE       P13
4350         ASSIGN      7,MX$CSIZE(P12,1)
4360 CARTQ   ASSIGN      17,MX$ITEM(P12,1)
4370         ASSIGN      8,0
4380         ASSIGN      9,1
4390 PARTQ   ASSIGN      8+,2
4400         ASSIGN      9+,2
4410         ASSIGN      5,MX$ITEM(P12,P8)
4420         ASSIGN      10,MX$PART(P5,1)
4430         ASSIGN      20,MX$ITEM(P12,P9)
4440         QUEUE       P10
4450         TRANSFER    SBR,TAKEP,RTRN2
4460         DEPART      P10
4470         LOOP        17,PARTQ
4480         LOOP        7,CARTQ
4490 FAC     SEIZE       P13
4500         ADVANCE     V*14
4510         ADVANCE     V$MTIME
4520 MTIME   FVARIABLE   V*16#MX$CSIZE(P12,1)
4530         DEPART      P13
4540         RELEASE     P13
4550         TRANSFER    P,RTRN3,1
```

Figure 6. GPSS code for manufacturing cell

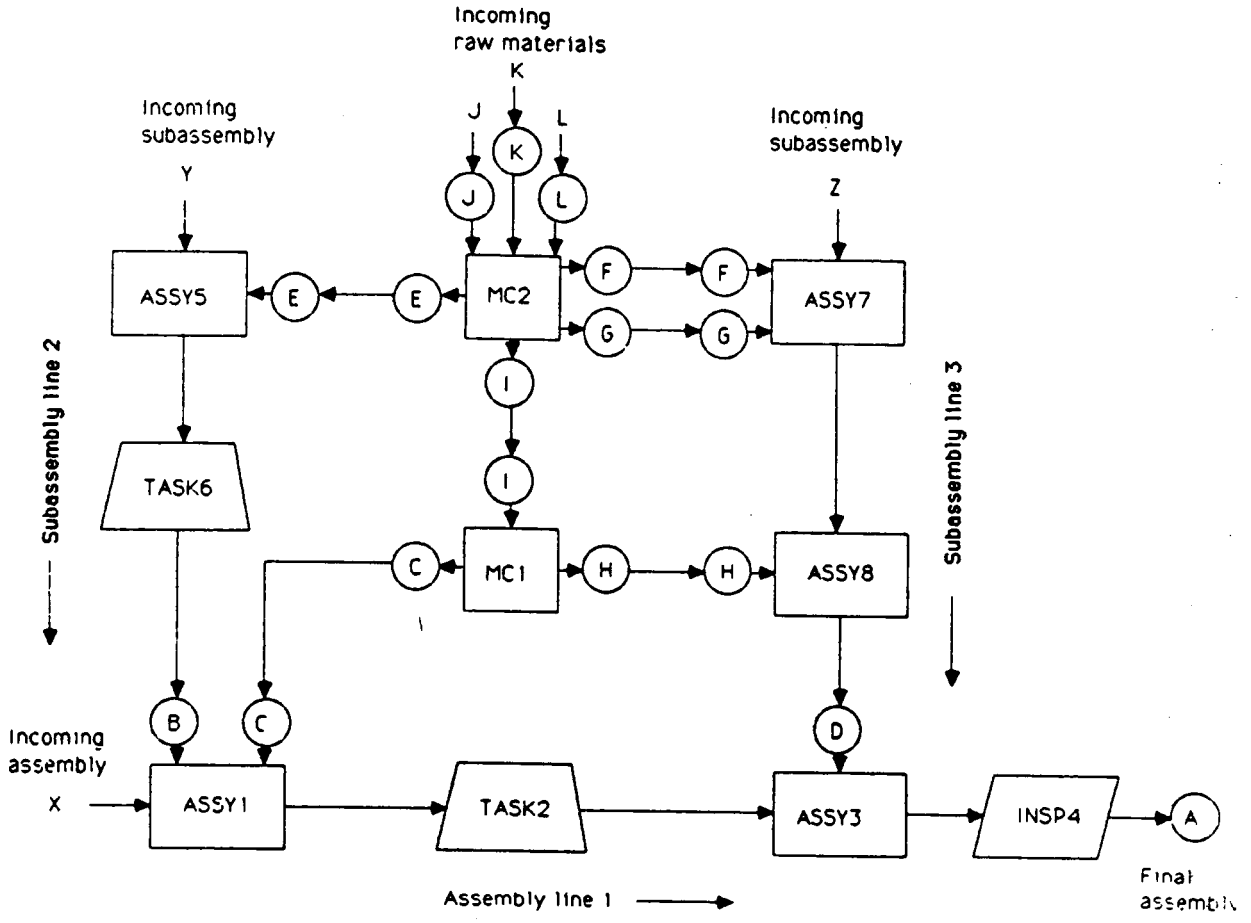

Figure 7. Manufacturing system

located throughout the manufacturing system.

Figure 8 is a partial listing of the interface dialogue between the system and the user to define the manufacturing system in Figure 7. Only the dialogue for subassembly line 2 and the part specification for parts B, C, and E is included in Figure 8.

Figure 9 is a partial listing of the GPSS code generated by the AMPS system. Lines 2770-3100 are the GPSS code for the assembly and two subassembly lines. Line 2820 is the transfer to the assembly station subroutine ASM. Line 2840 is the transfer to the task subroutine TASK. Line 2880 is the transfer to the inspection station subroutine INSP.

The GPSS program for the manufacturing system in Figure 7 consists of 344 blocks, of which 110 blocks were for the five macros, 25 blocks were for the main program and 209 blocks were matrix savevalues for defining the system attributes.

5.4 AMPS Summary

The Automatic Manufacturing Programming System (AMPS) has been used to successfully model several manufacturing systems. One of these systems has been a flexible manufacturing system (FMS) consisting of an 18 station cell and 9 alien stations (Schroer 1988). The FMS makes four different parts with each part requiring 47, 31, 22, and 22 operations respectively. The system was sufficiently different that another library of GPSS macros was written for the FMS stations. Consequently, the interactive user dialogue could not be used; however, the structured GPSS coding as shown in Figure 9 resulted in the FMS being modeled quickly and accurately.

The AMPS system has been successfully ported to a Texas Instrument Explorer workstation. This system is currently installed and under eva-

```
**********************
*   Create a line    *
**********************

. Name of line: y
. Number of stations: 2
3. Source of line:
     Type (see menu for selection): Beginning node
     Distribution: Normal
     Mean: 100
     Standard deviation: 5
4. Destination of line: Stock point
5. Name of the product of the line y: b

 you want to modify the input above?(Y or N) No.

station 5
 (1) Station id: 5
 (2) Type of station: Assembly station
 (3) Station name: assy5
 (4) Part required:
     Number of part types required: 1

       Name of part: e
       Number of part: 2
 (5) Time:
     Distribution: Normal
     Mean: 100
     Standard deviation: 5

Do you want to modify the input above?(Y or N) No.

station 6
 (1) Station id: 6
 (2) Type of station: Task station
 (3) Station name: task6
 (4) Time:
     Distribution: Normal
     Mean: 100
     Standard deviation: 5

Do you want to modify the input above?(Y or N) No.
End of line y
 more line to create? (Y or N) Yes.


**************************
*  Part specification    *
**************************

Part B
1. Part-id: 2
   Part-name: PA_B
   Supply-system: Push
   Capacity and initial inventory at the stock points:

   Maximum number of parts at stock point: 2000

   you want to modify the input above?(Y or N) No.

art C
   Part-id: 3
   Part-name: PA_C
   Supply-system: Pull from an inside source
. Capacity and initial inventory at the stock points:

   Mximum cart capacity (max. number of parts per cart): 10

   Current cart capacity (number of parts per cart): 4

   Maximum number of carts at demand stock point: 10

   Initial number of carts at demand stock point: 4

   Maximum number of carts at supply stock point: 10

   Initial number of carts at supply stock point: 4

5. Vehicle used to move carts between stock points:
     Name: truck1
     Time:
     Distribution: Uniform
     Minimum: 8
     Maximum: 12
6. Source-where the part is made:
     (1) Manufacturing cell: mc1
7. Items required to make the part:
     Number of item types required: 1

       Name of item: i
       Number of item: 2
8. Set up time for each cart:
     Distribution: Constant
     Constant: 0
9.Time to make a part:
     Distribution: Normal
     Mean: 30
     Standard deviation: 3

Do you want to modify the input above?(Y or N) No.

Part E
1. Part-id: 5
2. Part-name: PA_E
3. Supply-system: Pull from an inside source
4. Capacity and initial inventory at the stock points:

   Mximum cart capacity (max. number of parts per cart):

   Current cart capacity (number of parts per cart): 4

   Maximum number of carts at demand stock point: 10

   Initial number of carts at demand stock point: 4

   Maximum number of carts at supply stock point: 10

   Initial number of carts at supply stock point: 4

5. Vehicle used to move carts between stock points:
     Name: truck1
     Time:
     Distribution: Uniform
     Minimum: 8
     Maximum: 12
6. Source-where the part is made:
     (1) Manufacturing cell: mc2
7. Items required to make the part:
     Number of item types required: 2

       Name of item: j
       Number of item: 2

       Name of item: k
       Number of item: 1
8. Set up time for each cart:
     Distribution: Constant
     Constant: 0
9.Time to make a part:
     Distribution: Normal
     Mean: 10
     Standard deviation: 1

Do you want to modify the input above?(Y or N) No.
```

Figure 8. Partial AMPS interactive user interface dialogue

```
2770 ****************************************
2780 *      ASSEMBLY LINE x
2790 ****************************************
2800          GENERATE      V$TIME1
2810          ASSIGN        2,1
2820          TRANSFER      SBR,ASM,RTRN1
2830          ASSIGN        2,2
2840          TRANSFER      SBR,TASK,RTRN1
2850          ASSIGN        2,3
2860          TRANSFER      SBR,ASM,RTRN1
2870          ASSIGN        2,4
2880          TRANSFER      SBR,INSP,RTRN1
2890          ENTER         PA_a,1
2900          TERMINATE
2910 ****************************************
2920 *      ASSEMBLY LINE y
2930 ****************************************
2940          GENERATE      V$TIME5
2950          ASSIGN        2,5
2960          TRANSFER      SBR,ASM,RTRN1
2970          ASSIGN        2,6
2980          TRANSFER      SBR,TASK,RTRN1
2990          ENTER         PA_b,1
3000          TERMINATE
3010 ****************************************
3020 *      ASSEMBLY LINE z
3030 ****************************************
3040          GENERATE      V$TIME6
3050          ASSIGN        2,7
3060          TRANSFER      SBR,ASM,RTRN1
3070          ASSIGN        2,8
3080          TRANSFER      SBR,ASM,RTRN1
3090          ENTER         PA_d,1
3100          TERMINATE
```

Figure 9. Partial GPSS listing for assembly lines

luation at the School of Engineering, Auburn University.

In addition, the AMPS system has been submitted to NASA COSMIC through the NASA/MSFC Technology Utilization Office. The COSMIC ID number is MFS#28367. A user's manual, UAH Research Report No. 720, was written and accompanied the submission to COSMIC. The AMPS system has been documented in UAH Research Report No. 659.

6.0 AUTOMATIC NETWORK PROGRAMMING SYSTEM

6.1 Introduction

Large simulation projects have been undertaken for the space program. Many of these projects have the need for automated tools and techniques to support the model development. One of the projects involve simulating the countdown sequence prior to spacecraft liftoff. A number of constraints exist for launching a spacecraft into orbit. Many of these constraints are dependent on the mission. For example, on a lunar mission, these constraints may include allowable launch azimuth, required earth orbit inclination, daylight at the lunar landing area, and daylight at the primary recovery area. As a result of these constraints, a launch window of only several hours could exist during three consecutive days in a month.

Another constraint is the cryogenic propellents. The handling of the cryogenic propellents prevent a launch hold from one day to the next. For example, a launch that is scrubbed after the cryogenics have been loaded is generally delayed at least until the third day within the launch opportunity. In addition, a typical prelaunch consists of thousands of events, both on the launch vehicle, as well as the ground support equipment, that must be successfully completed to launch within a given launch

window.

The objective of the research presented is to develop an automatic programming system to assist the modeler of prelaunch countdown sequences define the problem, and to then automatically generate the program code in the target simulation language GPSS/PC. The AP system is called Automatic Network Programming System (ANPS). The domain of problems that can be solved by ANPS is the prelaunch activities of space vehicles and the operation of supporting ground support equipment. A broader domain is reliability network models of hardware systems and subsystems.

6.2 Previous Research

Synder et al. (1967) have developed a simulation model of the Saturn V prelaunch activities beginning at T-24 hours and continuing through T-0 hours, or lift-off. This model was used to predict the probability of launching the spacecraft within a given launch window. A second objective of the model was to identify locations in the countdown for placing holds and to determine the length of these holds. The model consisted of over 1100 vehicle subsystems and 400 ground support subsystems. A detailed time line was developed showing the interrelationships of these subsystems. In addition to the time line, the model input included operational data, reliability data, and maintenance data. The model was written in GPSS-II and ran on an IBM 360 computer.

The Synder model was expanded to include multiple launch windows and the operational sequence when a launch window was missed and the spacecraft had to be recycled to the next launch window (Schroer 1969). The model was used to predict the probability of launching a spacecraft within a given set of back-to-back launch windows. A second objective was to predict the probability of launching in a subsequent window, given a window had been

missed and a recycle sequence and a possible hold had to be executed before resuming the countdown.

The expanded model included two countdown sequences. The first sequence was the main countdown sequence identical to the Synder model. The second sequence was the recycle sequence that consisted of a number of backout sequences containing those events that were required to return the countdown to some preceding point. The recycle sequence also consisted of a recycle hold containing those activities that were required to sustain the vehicle status at a particular time in the countdown. The model was written in GPSS-II, contained 2300 blocks, several Fortran help routines and ran on the IBM 360 computer.

6.3 ANPS System Overview

Figure 10 gives an overview of the ANPS system. The ANPS system is designed using the elements of automatic programming as its foundation. The three AP elements in ANPS are an interactive user dialogue interface, a library of software modules, and an automatic simulation code generator. In Figure 10, the traditional programming task of flow charting has been replaced by the interactive user dialogue interface that results in the problem specification. Likewise, the program writing task has been replaced by the automatic code generator and the library of software macros, resulting in the GPSS code.

The ANPS system has four options available to the user. The input problem option begins the interactive user dialogue to define the problem specification. The edit problem option allows the user to edit an existing model by paging through the problem specification. The save problem option saves the problem specification on disk. The write GPSS option is selected when the user is satisfied with the problem specification. This option

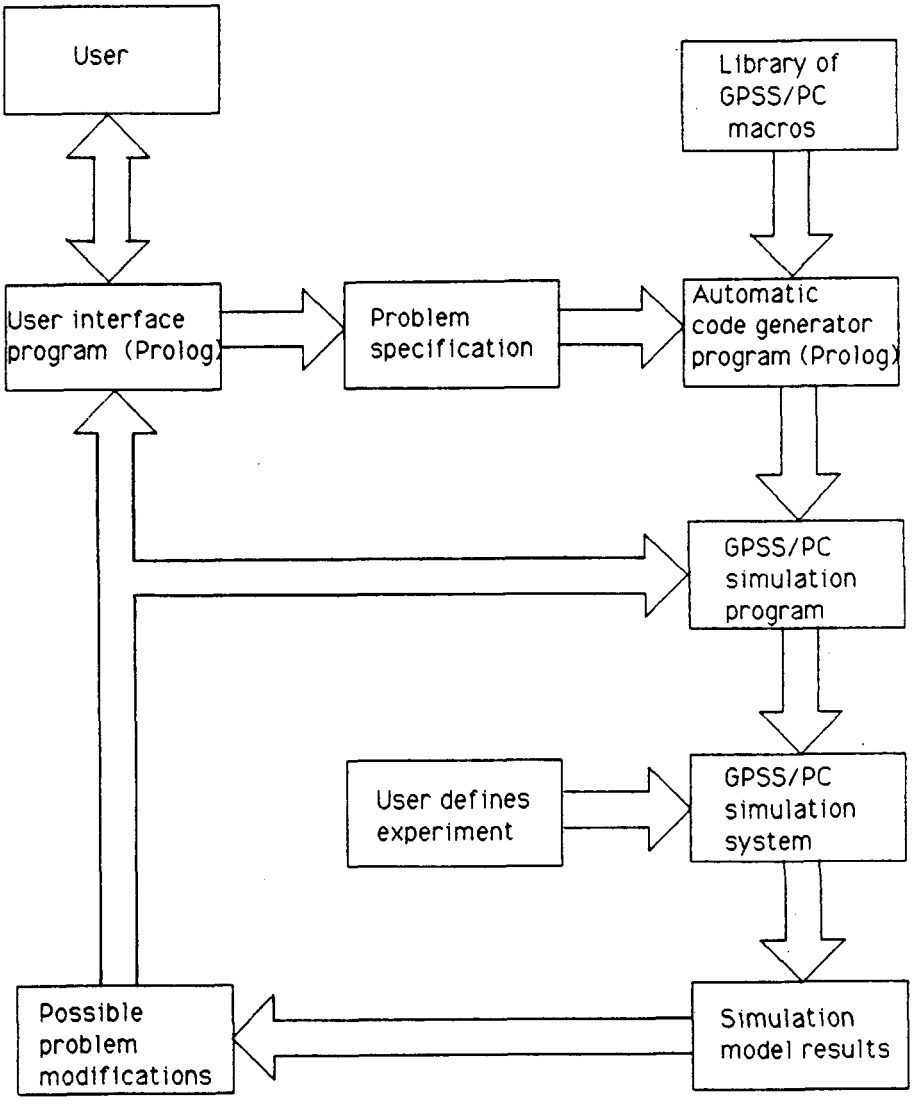

Figure 10. ANPS system overview

causes the ANPS system to automatically write the GPSS simulation code.

6.4 Interactive User Dialogue Interface

The ANPS system uses an interactive dialogue interface to assist the user define the problem specification. Using this interface, the user sits at a personal computer and enters into a dialogue with the ANPS system. Based on the user's responses, the interactive interface creates an internal problem specification file. This file includes the time line for the countdown sequence, the attributes for the activities, and the dependent relationships between the activities.

6.5 Library of GPSS Macros

The robustness of an AP system is dependent on the diversity and completeness of its library of software modules. Furthermore, this library is generally domain specific. When new modules or subroutines are needed, expert simulation programmers are needed to write the simulation code and to assure the proper interface.

Since the ANPS system is domain specific to prelaunch countdown sequences, the number of needed software modules is minimal. At this point of development, ANPS consists of the following four GPSS modules:

- Fixed activity operation function (VENT_A)
- Continuous activity operation function (VENT_B)
- Activity failure function (FAIL)
- Activity interrupt function (XACT_DELAY)

These modules were selected based on a detailed evaluation of the two previously discussed models by Synder (1967) and Schroer (1969). Interestingly, several of these previously developed modules were written as Fortran HELP routines using the old GPSS-II.

The fixed activity operation function (VENT_A) simulates the opera-

tion of each fixed time activity and its time to failure. If the activity fails during its operation, the transaction is forwarded to the activity failure function (FAIL).

The continuous activity operation function (VENT_B) simulates the operation of each continuous time activity and its time to failure. This activity is not completed until all other related activities are completed. For example, system power is a continuous time function that will be on until all activities requiring power are completed. If the activity fails, the transaction is forwarded to the activity failure function (FAIL).

The activity failure function (FAIL) simulates the failure of an activity as indicated by functions VENT_A and VENT_B. When an activity fails, all the dependent activities enter a hold state. The function then simulates the time to repair the activity. If another activity fails during the delay of a dependent activity and the dependent activity is dependent on the first failed activity, the additional time to repair, if any, is added to the delay of the dependent activity. The failure function assumes that a dependent activity that has been delayed cannot fail during the delay. The activity interrupt function XACT_DELAY contains the logic to add any additional time to an activity on hold if another activity fails during the hold and the held activity is dependent on the failed activity.

Figure 11 is a listing of the GPSS code for the continuous activity function VENT_B. Note that the subroutine makes extensive use of indirect addressing. The system also contains a large number of matrix savevalues for transferring data between the subroutines and the main program. Initially, all the input data from the problem specification are entered into these matrix savevalues.

6.6 Automatic Simulation Code Generator

```
1830 *
1832 *       CONTINUOUS ACTIVITY TIME SIMULATION GENERATOR
1834 *
1840 VENT_B SEIZE          P2
1842       ASSIGN          98,MX$F_TIME(P3,1)
1843       SAVEVALUE       FTS,V*98
1845 TIME9  FVARIABLE      X$FTS
1850 TIME8  FVARIABLE      X$FTS/100
1855       TEST L          V$TIME9,100,BACK6
1860       ASSIGN          TIM3,1
1865       ASSIGN          BSUM,V$TIME9
1870       TRANSFER        ,BACK5
1875 BACK6 ASSIGN          TIM3,V$TIME8
1880       ASSIGN          BSUM,100
1885 BACK5 ASSIGN          NR_LOOPS,P$BSUM
1890 BACK4 GATE LR         MX$SWITCH1(P3,1),ENDA
1895       ADVANCE         P$TIM3
1900       LOOP            NR_LOOPS,BACK4
1905       ASSIGN          ROW,P3
1910       TRANSFER        SBR,FAIL,RTRN1
1915       TRANSFER        ,BACK5
1920 ENDA  RELEASE         P2
1925       TRANSFER        P,RTRN2,1
```

Figure 11. Continuous activity macro

The output from the interactive dialogue interface, or the problem specification, is used as input by ANPS to the automatic code generator program. The code generator program selects the appropriate macros from the GPSS library and then generates the simulation code in the target language GPSS/PC.

The output of the code generator program is a GPSS/PC program file. The experimental frame, such as the run statements, are then added and the GPSS program executed. The output file is stored on a diskette or printed on the personal computer. To change the GPSS model, the user must recall the problem specification file. The interface program provides the user with a number of options to change or modify the problem specification. The code generator then rewrites the GPSS/PC program. The GPSS model can also be changed by using the standard text editor within Turbo Prolog.

The ANPS system is written in Turbo Prolog (Borland 1986) for the IBM class of personal computer. The library of macros is written in GPSS/PC. ANPS contains 1218 lines of Prolog code and 86 subroutines. The simulation code generated by ANPS is GPSS/PC (Minuteman 1986).

6.7 System Constraints

The ANPS macros imposed the following constraints:

- An activity failure will cause that activity to be delayed until the failure has been repaired.
- All dependent activities will also be delayed for the same time until the failure has been repaired.
- If another activity fails during the delay of a dependent activity and the dependent activity is also dependent on the just failed activity, the additional time to repair, if any, is added to the delay of the dependent activity.

- A dependent activity that has been delayed cannot fail during the delay time and will not cause other dependent activities to be delayed.
- No two continuous activities can end on the same node.
- No two activities can start from the same node and terminate on the same node.

6.8 Sample Problem

Figure 12 is a time line for a simplified prelaunch countdown sequence consisting of 16 fixed activities and two continuous activities. Figure 13 is the time line redrawn in the form of a network diagram and structured for input to the ANPS system. The dotted lines in these figures indicate time line constraints. For example, activities ACT11 and ACT15 must be completed before starting activity ACT12. ACT21 is a dummy activity with zero time that is used to impose the activity ACT15 constraint.

Several other dummy activities were also required to construct the network diagram. For example, dummy activity ACT23 was added to simulate the termination of the second continuous activity ACT2, since no more than one continuous activity can end at a node. Also, dummy activity ACT19 was added at the completion of activity ACT5 since no two activities can start from the same node, node 2, and end at the same node, node 4.

Table II contains the time attributes for the activities in the prelaunch countdown. These attributes include activity duration, activity time to failures, and activity time to repairs. Note that activities ACT1 and ACT2 have continuous operation times. That is, these activities will operate during the entire prelaunch countdown. An example of a continuous activity is electrical power that may be needed to operate a number of activities.

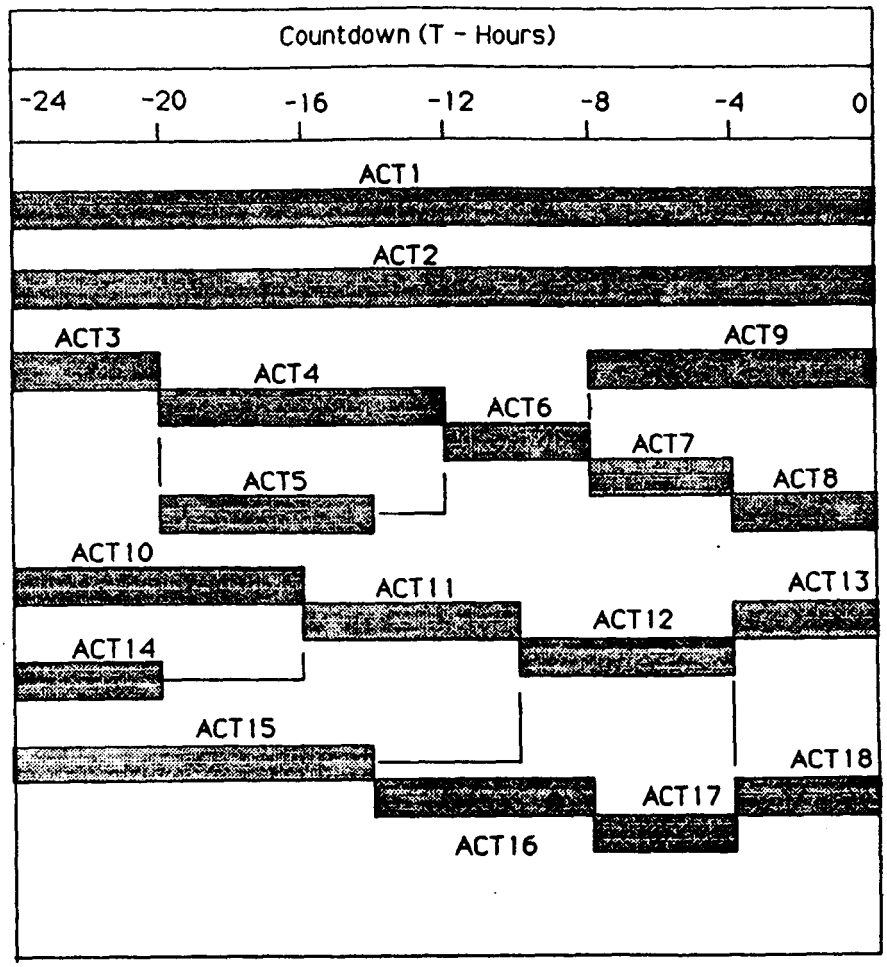

Figure 12. Prelaunch countdown for sample problem

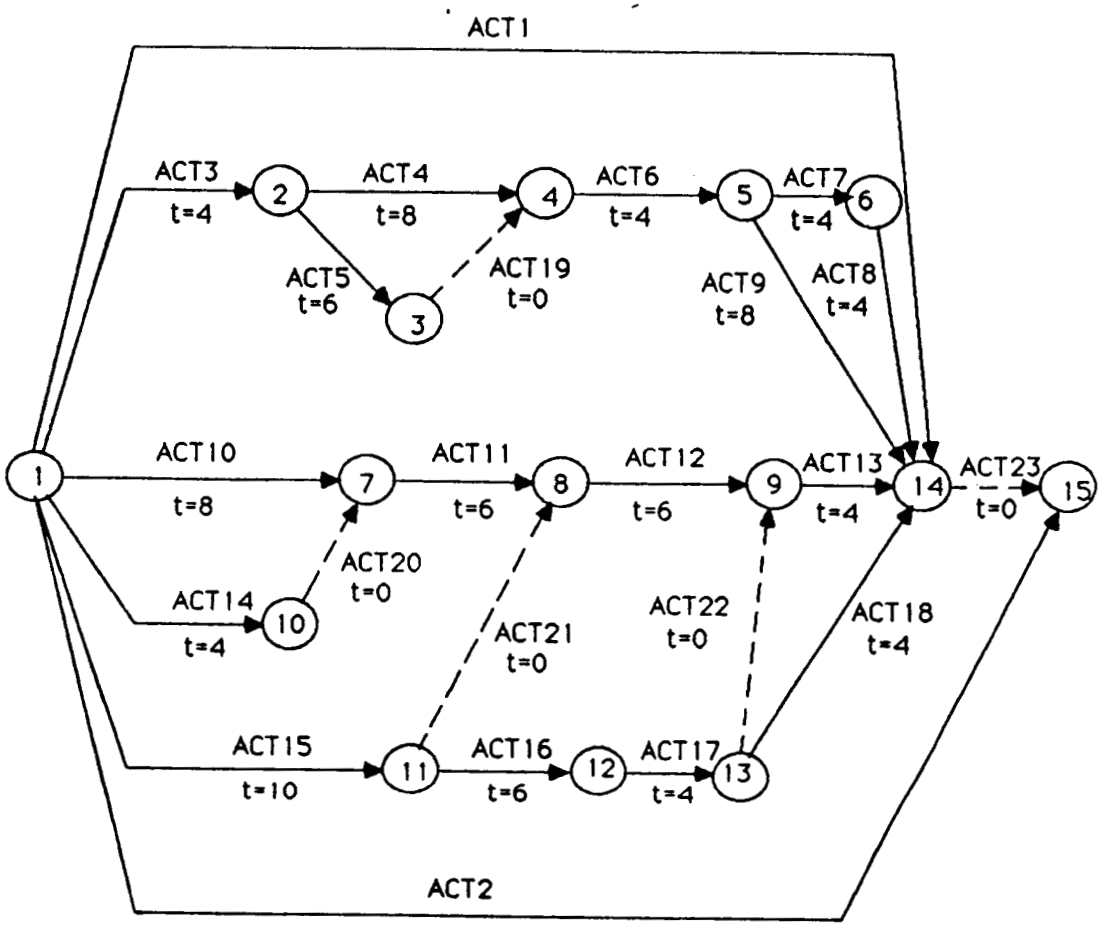

Figure 13. Network representation of prelauch sequence

Table II. Countdown sequence attributes

Activity	Duration (hours)	Failure time (hours)	Repair time (minutes)
1	Continuous	E(33)	N(60,6)
2	Continuous	E(33)	N(60,6)
3	4	E(12)	N(30,3)
4	8	E(12)	N(30,3)
5	6	E(12)	N(60,6)
6	4	E(12)	N(45,5)
7	4	E(12)	N(45,5)
8	4	E(12)	N(60,6)
9	8	E(12)	N(60,6)
10	8	E(12)	N(60,6)
11	6	E(12)	N(45,5)
12	6	E(12)	N(30,3)
13	4	E(12)	N(60,6)
14	4	E(12)	N(90,9)
15	10	E(12)	N(60,6)
16	6	E(12)	N(120,2)
17	4	E(12)	N(60,6)
18	4	E(12)	N(45,5)
19	Dummy	-	-
20	Dummy	-	-
21	Dummy	-	-
22	Dummy	-	-
23	Dummy	-	-

Table III. Operational dependencies between activities

	Dependent Activity																						
Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	x		x	x	x							x	x					x					
2		x					x	x	x					x	x								
3			x																				
4				x	x																		
5					x																		
6						x																	
7							x																
8								x					x										
9									x														
10										x													
11											x												
12						x						x											
13													x										
14														x									
15															x								
16						x					x					x							
17																	x						
18													x					x					
19																			x				
20																				x			
21																					x		
22																						x	
23																							x

Table III contains the operational dependencies between the activities. In other words, the table gives the effect of an activity failure on other activities in the prelaunch. For example, a failure of the continuous activity ACT1 will cause a stopping of activities ACT3, ACT4, ACT5, ACT12, ACT13, and ACT18. Likewise, a failure of activity ACT4 will cause a stopping of activity ACT5.

The interactive dialogue between the user and the ANPS system consisted of 23 screens with one screen for each activity. Figure 14 is a screen dump for defining activity ACT4. ACT4 starts at node 2 and ends at node 4. The activity type is a fixed duration and the operation time is eight hours. The time to failure follows the exponential distribution with a mean of 12 hours. The time to repair follows the normal distribution with a mean of 30 minutes and a standard deviation of three minutes. A failure of activity ACT4 will result in a stopping of activity ACT5.

At the completion of the problem definition, the ANPS system automatically generates the corresponding GPSS code. Figure 15 is a partial listing of the main GPSS program that was generated by the ANPS system. The complete main program consists of 183 GPSS blocks. This program consists of a series of ASSIGN, SPLIT, TRANSFER and ASSEMBLE blocks.

For example, line 2048 is the start of activity ACT3. Line 2053 ends activity ACT3. Line 2049 splits another transaction to start activity ACT10. The transactions are sent to one of two subroutines depending on the activity type. For example, in line 2039 the transaction is routed to the continuous activity subroutine VENT_B. Other transactions, such as in lines 2052 and 2058, are routed to the fixed activity subroutine VENT_A. These subroutines simulate the activity time and transfer the transactions to the failure subroutine FAIL which simulates the time to failures and

```
     Name for GPSS Program              : EX1.23

 1. Number of activities                : 23
 2.  Activity attributes:
     Activity name                      : $ACT4
     Activity type (fixed/variable)     : FIXED
     Duration distribution type         : CONSTANT
              mean time                 : 480

     Starting node number               : 2
     Ending   node number               : 4
     MTTF     distribution type         : EXPONENTIAL
              mean time                 : 720

     MTTR     distribution type         : NORMAL
              mean time                 : 30
              standard deviation        : 3
     Number of dependent activities     : 1

  Do you want to modify the input above ? (Y/N) : N
```

Figure 14. Interactive dialogue for defining activity ACT4

```
1945          GENERATE      ,,,1
1950 MORE     SPLIT         1,MM
1955          GATE LS       SWITCH_MORE
1960          LOGIC R       SWITCH_MORE
1965          TRANSFER      ,MORE
1970 MM       MARK          SYSTIME
2000 EV1      ADVANCE
2001          TRANSFER      ,A1
2002 EV2      ADVANCE
2003          TRANSFER      ,A4
2004 EV3      ADVANCE
2005          TRANSFER      ,A19
2006 EV4      ASSEMBLE      2
2007          TRANSFER      ,A6
2008 EV5      ADVANCE
2009          TRANSFER      ,A7
2010 EV6      ADVANCE
2011          TRANSFER      ,A8
2012 EV7      ASSEMBLE      2
2013          TRANSFER      ,A11
2014 EV8      ASSEMBLE      2
2015          TRANSFER      ,A12
2016 EV9      ASSEMBLE      2
2017          TRANSFER      ,A13
2018 EV10     ADVANCE
2019          TRANSFER      ,A20
2020 EV11     ADVANCE
2021          TRANSFER      ,A16
2022 EV12     ADVANCE
2023          TRANSFER      ,A17
2024 EV13     ADVANCE
2025          TRANSFER      ,A18
2026 EV14     ASSEMBLE      4
2027          LOGIC S       SWITCH_END1
2028 EEV14    ASSEMBLE      2
2029          TRANSFER      ,A23
2030 EV15     ASSEMBLE      1
2031          LOGIC S       SWITCH_END2
2032 EEV15    ASSEMBLE      2
2033          TRANSFER      ,END1
2034 A1       ASSIGN        2,$ACT1
2035          SPLIT         1,A2
2036          ASSIGN        3,1
2038          LOGIC R       SWITCH_END1
2039          TRANSFER      SBR,VENT_B,RTRN2
2040          TRANSFER      ,EEV14
2041 A2       ASSIGN        2,$ACT2
2042          SPLIT         1,A3
2043          ASSIGN        3,2
2045          LOGIC R       SWITCH_END2
2046          TRANSFER      SBR,VENT_B,RTRN2
2047          TRANSFER      ,EEV15
2048 A3       ASSIGN        2,$ACT3
2049          SPLIT         1,A10
2050          ASSIGN        3,3
2052          TRANSFER      SBR,VENT_A,RTRN2
2053          TRANSFER      ,EV2
2054 A10      ASSIGN        2,$ACT10
2055          SPLIT         1,A14
2056          ASSIGN        3,10
2058          TRANSFER      SBR,VENT_A,RTRN2
2059          TRANSFER      ,EV7
2060 A14      ASSIGN        2,$ACT14
2061          SPLIT         1,A15
2062          ASSIGN        3,14
2064          TRANSFER      SBR,VENT_A,RTRN2
2065          TRANSFER      ,EV10
2066 A15      ASSIGN        2,$ACT15
2067          ASSIGN        3,15
2069          TRANSFER      SBR,VENT_A,RTRN2
2070          TRANSFER      ,EV11
2071 A4       ASSIGN        2,$ACT4
2072          SPLIT         1,A5
2074          ASSIGN        3,4
2076          TRANSFER      SBR,VENT_A,RTRN2
2077          TRANSFER      ,EV4
2078 A5       ASSIGN        2,$ACT5
2079          ASSIGN        3,5
2081          TRANSFER      SBR,VENT_A,RTRN2
2082          TRANSFER      ,EV5
```

Figure 15. Partial GPSS listing of main program generated by ANPS

ORIGINAL PAGE IS
OF POOR QUALITY

time to repairs. After the activity times have been simulated, the transactions are transferred to the appropriate ASSEMBLE blocks. Once all the required transactions have been assembled at the ASSEMBLE block, the remaining branches in the network are simulated. For example, line 2006 assembles activities ACT4 and ACT19.

The complete GPSS program consists of:

- 409 blocks and statements for the entire program
- 183 blocks for the main program that define the network
- 62 blocks for the subroutines from the library of macros
- 83 matrix savevalues that define the variable statements for the activity times, failure times, and repair times
- 66 variable statements that contain the actual time expressions

Figure 16 gives the distribution of time to complete the prelaunch sequence in Figure 12. This distribution is based on the simulation of 200 launches. The mean time to complete the countdown is 34.2 hours. Launch vehicle availability (LVA) is defined as the probability of launching within a given launch window. The LVA for up to a six hour window is given in Figure 17. The LVA for a two hour window is 0.015 and increases to 0.596 for a six hour window.

6.9 ANPS Summary

The Automatic Network Programming System (ANPS) has the potential for use in rapid prototyping of reliability networks. Specific applications for ANPS are in modeling prelaunch activities of space vehicles, ground support equipment, space vehicle turn around plans, space transportation systems and operational planning, and hardware systems with multiple subsystems. The ANPS system has been documented in UAH Research Report No. 704.

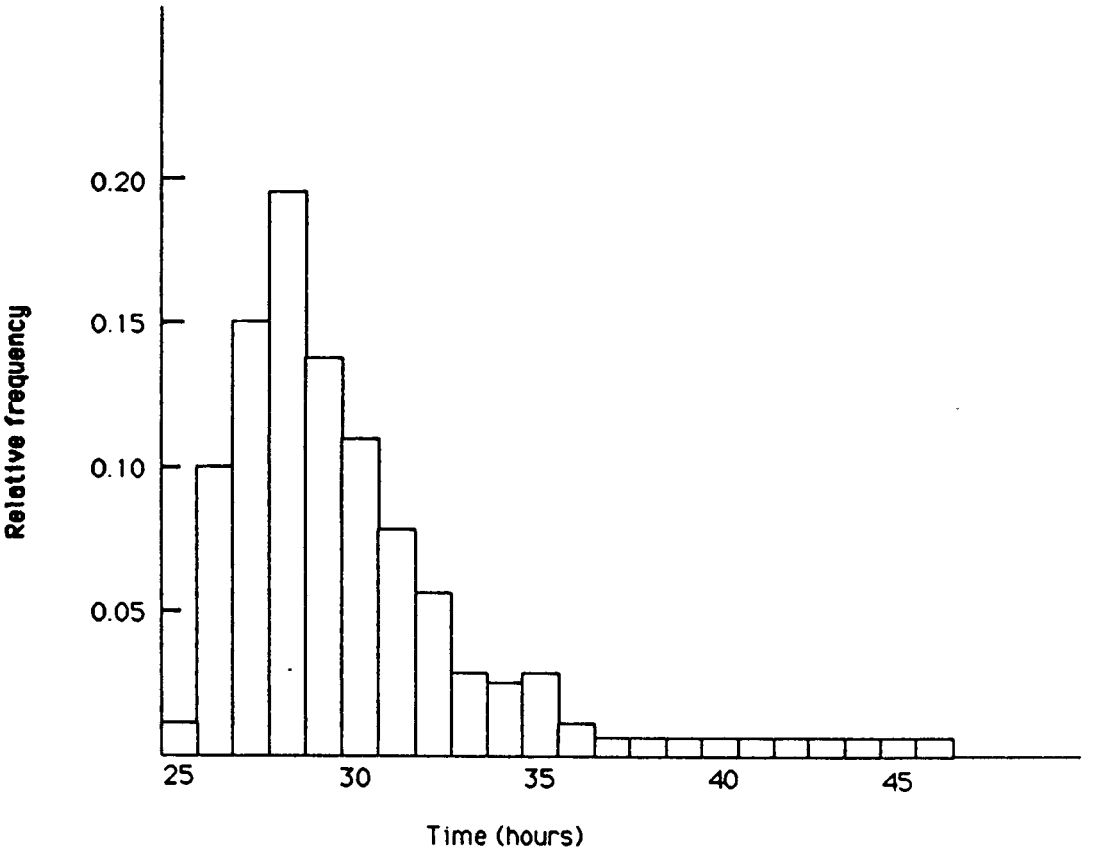

Figure 16. Time to complete prelaunch countdown

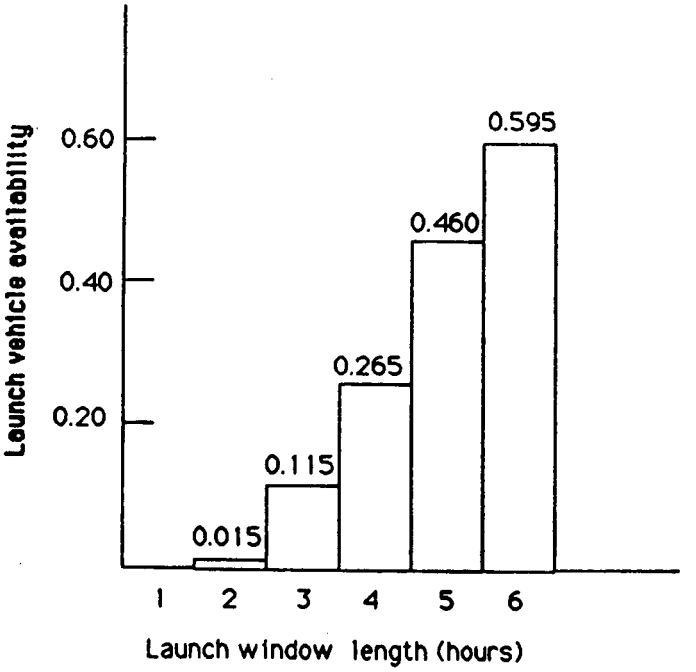

Figure 17. Launch vehicle availability

7.0 CONCLUSIONS

Imbedding AI within discrete event simulation is an effective and efficient method for modeling manufacturing systems and reliability networks. There are a number of potential advantages of automatic programming such as AMPS and ANPS. These advantages include:

- Improved Clarity - The GPSS code generated by AMPS and ANPS is a structured simulation code that is easy to read, trace and modify.
- Increased Productivity - The AMPS and ANPS systems should result in a significant increase in the lines of simulation code written per hour.
- Rapid Prototyping - Given the availability of the necessary macros, the system permits rapid prototyping. In addition, the system produces executable simulation code that is syntax error free.
- Easier Maintenance - The structured approach minimizes the effort required in locating errors and making program modifications.
- Reduced Modeler Knowledge - Hopefully the modeler's knowledge of the target simulation language should be reduced.

The AMPS and ANPS systems use the interactive user dialogue to assist the modeler define the problem specification. The interactive user interface:

- Provides for a structured procedure for acquiring information on the system being modeled.
- Expedites the definition of the problem specification.
- Assures a complete and detailed definition of the problem specification.

In contrast to the advantages, there are also several disadvantages. One disadvantage, and probably the most significant, is that the AMPS and ANPS system are domain specific. Another problem is that similar domains may require additional new macros or subroutines. An experienced GPSS simulationist must then be used to write the code for these macros. Another disadvantage is that the AP systems require more memory and execution time than a nonstructural equivalent program. However, this disadvantage is not as significant as in prior years because computers are becoming faster and have more memory. A fourth disadvantage is the user attitude problem of learning something new and different.

In summary, the AMPS and ANPS systems are still in the research stages of development. Hopefully, some day these types of automatic programming techniques will move from the research stages to actual implementation and operational use by the end user.

The following is a list of publications on the research supported by NASA Grant NAG8-641.

1. "Use of Simulation Generators in Modeling Manufacturing Systems," F. T. Tseng and B. J. Schroer, Proceedings Southeastern Computer Simulation Conference, October 1987, Huntsville, AL, pp. 149-153.
2. "LISP-Based Simulation Generators for Modeling Complex Space Processes," F. T. Tseng, B. J. Schroer, and W. S. Dwan, Proceedings 3rd Conference on Artificial Intelligence for Space Applications, November 1987, Huntsville, AL, pp. 243-247.
3. "Modeling Complex Manufacturing Systems Using Simulation," B. J. Schroer and F. T. Tseng, Proceedings 1987 Winter Simulation Conference, December 1987, Atlanta, GA, pp. 677-682.

4. A Simulation Assistant for Modeling Manufacturing Systems, B. J. Schroer, F. T. Tseng, and W. S. Dwan, UAH Research Report No. 659, January 1988.

5. "Constructing Discrete Event Models in GPSS Using a Simulation Assistant," F. T. Tseng and B. J. Schroer, ORSA/TIMS, Washington, D. C., May 1988.

6. "Automatic Programming Systems (AMPS), B. J. Schroer, F. T. Tseng, and J. W. Wolfsberger, Proceedings for Conference on Space and Military Applications of Automation and Robotics, Huntsville, AL, June 1988.

7. "Automatic Programming of Manufacturing Simulation Models," B. J. Schroer, F. T. Tseng, S. X. Zhang, and J. W. Wolfsberger, Proceedings 1988 Summer Computer Simulation Conference, Seattle, WA, July 1988, pp. 569-574.

8. "Modeling Complex Manufacturing Systems Using Discrete Event Simulation," B. J. Schroer and F. T. Tseng, Computers and Industrial Engineering (accepted for publication).

9. Automatic Network Programming System (ANPS), F. T. Tseng, S. X. Zhang, and B. J. Schroer, UAH Research Report No. 704, June 1988.

10. "Using Automatic Programming for Simulating Reliability Network Models," F. T. Tseng, B. J. Schroer, S. X. Zhang, and J. W. Wolfsberger, Proceedings Fourth Conference on Artificial Intelligence for Space Applications, Huntsville, AL, November 15-16, 1988.

11. "Automatic Programming Assistant for Network Simulation Models," F. T. Tseng, B. J. Schroer, S. X. Zhang, and J. W. Wolfsberger,

Proceedings 1988 Winter Simulation Conference, December 12-14, 1988, San Diego, CA.

12. Automatic Manufacturing Programming System (AMPS) User's Manual, B. J. Schroer, F. T. Tseng, and W. S. Dwan, UAH Research Report No. 720, Semptember 1988.

8.0 REFERENCES

Barr, A. and E. A. Feigenbaum, 1982, The Handbook of Artificial Intelligence, Vol. 2, W. Kaufman, Inc., CA.

Brazier, M. K. and R. E. Shannon. 1987. "Automatic Programming of AGVS Simulation Models," 1987 Winter Simulation Conference, Atlanta, GA, (December) pp. 703 - 708.

Ford, D. R. and B. J. Schroer. 1987. "An Expert Manufacturing Simulation System." Simulation, Vol. 48, No. 5, (May) pp. 193-200.

GPSS/PC Reference Manual, 1986, Minuteman Software, Stow, MA.

Haddock, J. and R. P. Davis. 1985. "Building a Simulation Generator for Manufacturing Cell Design and Control." Annual International Industrial Engineering Spring Conference Proceedings, Los Angeles, CA, (May) pp. 237-244.

Heidorn, G. E. 1974. "English as a Very High Level Language for Simulation Programming." SIGPLAN Notices, Vol. 9, No. 4, pp. 91-100.

Khoshnevis, B. and A. P. Chen. 1986. "An Expert Simulation Model Builder." Intelligent Simulation Environment, Society for Computer Simulation, Vol. 17, No. 1, pp. 129-132.

Murray, K. J. and S. V. Sheppard. 1988. "Knowledge-based Simulation Model Specification," Simulation, Vol. 50, No. 3, (March) pp. 112-119.

Schroer, B. J. 1988. A Simulation Model of a Manufacturing Cell. University of Alabama in Huntsville, Research Report No. 676, April.

Schroer, B. J., F. T. Tseng and W. S. Dwan. 1988. A Simulation Assistant for Modeling Manufacturing Systems, University of Alabama in Huntsville Research Report No. 659, January.

Schroer, B. J., F. T. Tseng, and W. S. Dwan. 1988, Automatic Manufacturing Programming System (AMPS) User's Manual, UAH Research Report No. 720, September.

Schroer, B. J. 1969. "Saturn V Prelaunch Systems Simulation Model for a Launch Opportunity Containing Multiple Launch Windows," Proceedings Third Conference on Applications of Simulation, Los Angeles, December, pp. 503-511.

Synder, J. E., E. R. Bennich and Y. H. Lindsey. 1967. "Implementation of Advanced Simulation Techniques for Predicting the Saturn V Launch Vehicle System Behavior," Journal of Spacecraft and Rockets, Vol. 4, No. 8, pp. 998-1002.

Turbo Prolog 2.0 Reference Guide (1986) Borland International, Scotts Valley, CA.