Using Automatic Programming for Simulating Reliability Network Models

Fan T. Tseng
Bernard J. Schroer
University of Alabama in Huntsville
Huntsville, Alabama, USA 35899

S. X. Zhang
Northwestern Polytechnical University
Xian, Shaanxi, China

John W. Wolfsberger
NASA Marshall Space Flight Center
Marshall Space Flight Center, Alabama, USA 35812

ABSTRACT

This paper presents the development of an automatic programming system for assisting modelers of reliability networks define problems and then automatically generate the corresponding code in the target simulation language GPSS/PC.

INTRODUCTION

There has always been a desire of software developers to automate more and more of the computer programming process. The goal of these developers has been to have a system that can carry on a natural language dialogue with the user in defining his problem and then to automatically generate the appropriate computer code. The term automatic programming (AP) has been defined as an application of artificial intelligence (AI) in automating some aspects of the computer programming process (Barr and Feigenbaum 1982). This automation is generally accomplished by developing another program, an AP system, that raises the level of specifying computer program instructions. In other words, an AP system is a program that helps programmers write programs.

An AP system should improve the overall environment for defining and writing the program (Brazier and Shannon 1987). As a result of this improved environment, there should be a reduction in the amount of detail that the programmer needs to know. Quite possibly, the user could even do his own programming with the help of an AP system. Also, this improved environment should result in a more natural way for the user to define his problem that closely resembles the user's way of thinking and looking at problems.

RESEARCH GOAL

The goal of the research presented in this paper is to develop an AP system to assist the modeler of reliability networks

define problems, 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 include prelaunch activities of space vehicles, operation of ground support equipment, space vehicle turn around activities, space transportation systems and operational plans, and hardware system with multiple subsystems. The ANPS system requires that the problem be defined by:

- A network of activities with starting and stopping events.
- Activities with either fixed or continuous times.
- Activity failures and repairs (mean times to failure and repair).
- Operational dependencies between activities.

PREVIOUS RESEARCH

Synder et al. (1967) 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 launch vehicle availability (LVA). LVA was defined as 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 Synder 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 original 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.

A goal of the ANPS system is to be able to model these types of applications more quickly and accurately than previously done using conventional simulation techniques.

ANPS SYSTEM

Figure 1 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 1, the traditional programming task of flow charting has been replaced by the interactive user dialogue interface and the problem specification. Likewise, the program writing task in Figure 1 has been replaced by the automatic code generator and the library of software macros. The ANPS system is written in Turbo Prolog (Borland 1986) on an IBM PC class of personal computer. The system contains 1218 lines of code and 86 subroutines. The simulation code generated by ANPS is GPSS/PC (Minuteman 1986).

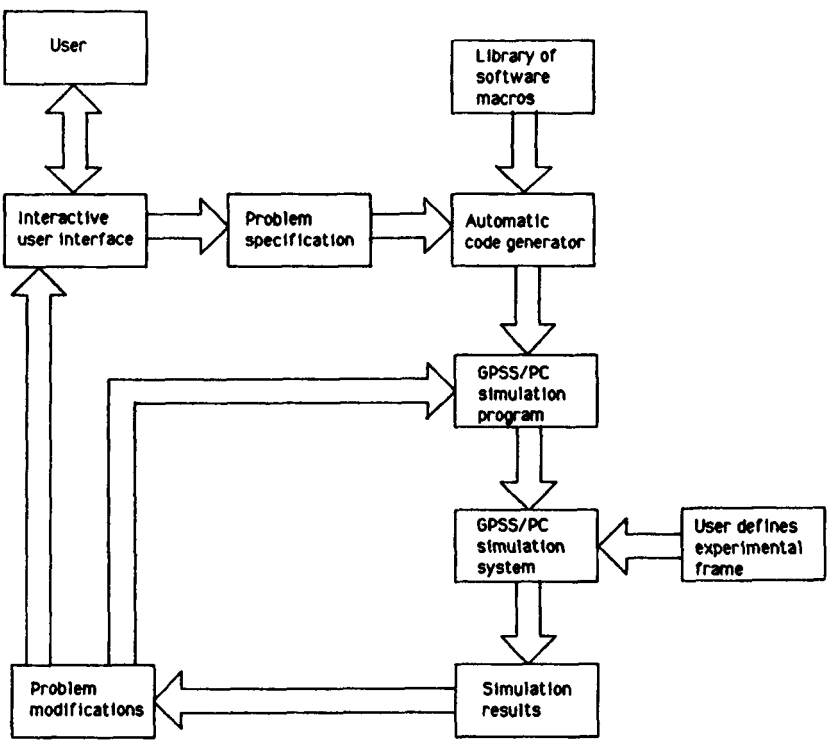

Figure 1. ANPS system overview

Interactive User Dialogue Interface

There are three commonly used AP user interfaces. These are a natural language interface, a graphical user interface, and an interactive dialogue interface. Several natural language interface developments are Heidorn (1974) and Ford and Schroer (1987). An example of a graphical interface development is Khoshnevis and Chen (1986). Several interactive dialogue interfaces are Haddock and Davis (1985), Brazier and Shannon (1987), and Murray and Sheppard (1988).

The ANPS system uses an interactive dialogue interface. This interface is probably the most common and easiest to develop interface. Using this interface, the user, or modeler, enters into a dialogue with the ANPS system to define the problem specification, or model.

Library of Software Modules

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 system reliability networks, the number of needed software modules is minimal. At this point of development, ANPS consists of the following four modules:

- Fixed activity operation function
- Variable activity operation function
- Activity failure function
- Activity interrupt function

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 operation 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 variable activity operation function (VENT_B) simulates the operation of each variable 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

variable 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 simu-lates 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 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 2 is a listing of the GPSS code for the fixed activity function VENT_A. 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.

Automatic Simulation Code Generator

The output from the interactive dialogue interface, or the problem specification, is then used as input to the code generator program which automatically writes the program code in the target simulation language GPSS. The system creates the main program that includes the appropriate calls to the selected library macros.

```
1360 *
1370 *          ACTIVITY TIME SIMULATION GENERATOR
1380 *
1390 VENT_A SEIZE          P2
1400        ASSIGN         ETIME,MX$WORK_TIME(P3,1)
1410 BACK3  ASSIGN         MTTF,MX$F_TIME(P3,1)
1420        TEST L         P$MTTF,P$ETIME,NOFAIL
1430        ADVANCE        P$MTTF
1440        ASSIGN         ROW,P3
1450        TRANSFER       SBR,FAIL,RTRN1
1460        ASSIGN         REST_TIME,V$TIME3
1470 TIME3  FVARIABLE      P$ETIME-P$MTTF
1480        ASSIGN         ETIME,P$REST_TIME
1490        TRANSFER       ,BACK3
1500 NOFAIL ADVANCE        P$ETIME
1510        RELEASE        P2
1520        TRANSFER       P,RTRN2,1
```

Figure 2. GPSS listing for function VENT_A

SAMPLE PROBLEM

Figure 3 is a time line for a typical network consisting of nine fixed activities and two variable activities. Figure 4 is the time line redrawn in the form of a network diagram. Activity 12 is a dummy activity with the time equal to zero.

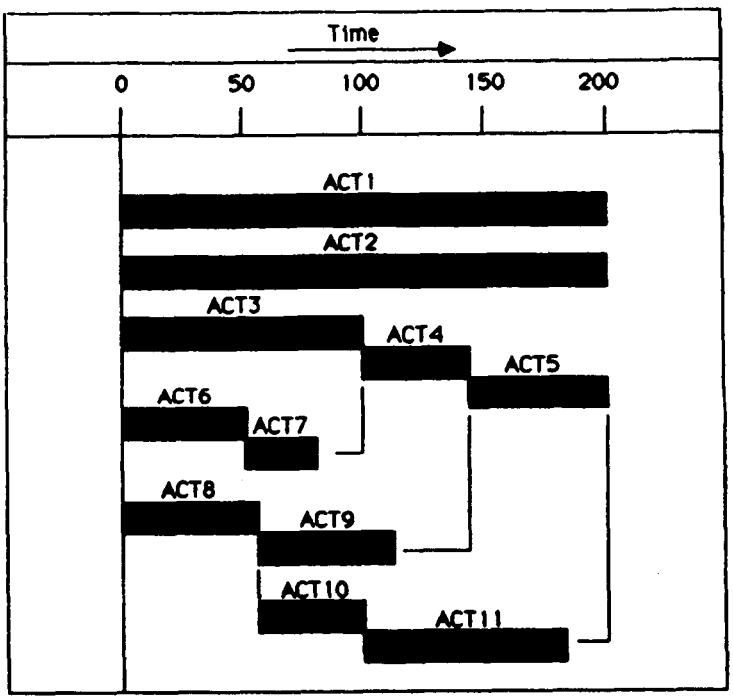

Figure 3. Sample problem time line

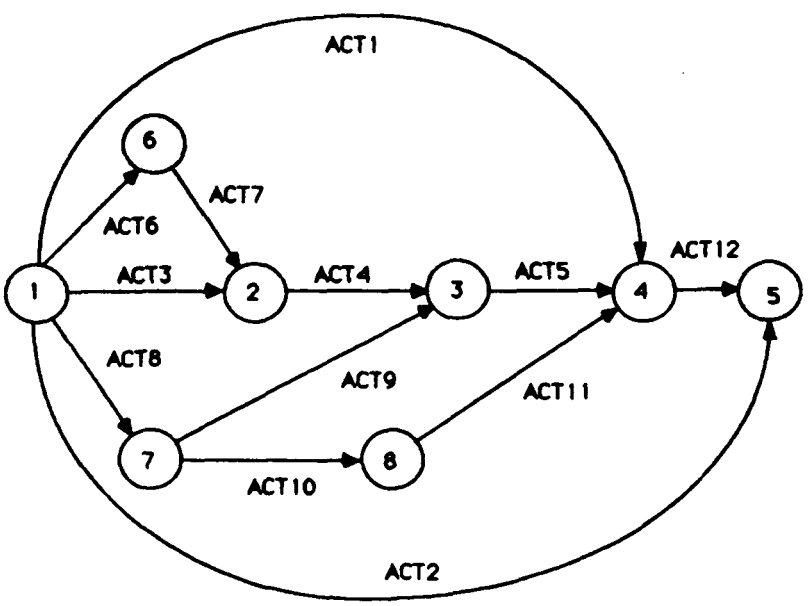

Figure 4. Sample problem reliability network

Table I gives the time parameters for the activities. These parameters include activity duration, time to failure and time to repair. Activities ACT1 and ACT2 have variable times; therefore, the activities will operate during the entire duration of the system. Table II gives the operational dependencies between the activities. For example, a failure of activity ACT1 will cause a stopping of activities ACT3, ACT4, and ACT5. Likewise, a failure of activity ACT9 will also stop activity ACT5.

Table I. Activity Time Parameters

Activity	Duration	Time to Failure	Time to Repair
ACT1	Variable	E(200)	N(20,4)
ACT2	Variable	E(200)	N(20,4)
ACT3	100	E(120)	N(10,2)
ACT4	40	E(60)	N(5,1)
ACT5	60	E(80)	N(5,1)
ACT6	50	E(60)	N(5,1)
ACT7	30	E(100)	N(10,2)
ACT8	60	E(100)	N(10,2)
ACT9	60	E(50)	N(5,1)
ACT10	40	E(60)	N(5,1)
ACT11	80	E(100)	N(10,2)
ACT12	Dummy	0	0

Table II. Operational Dependencies Between Activities

	Dependent Activities											
Activity	ACT1	ACT2	ACT3	ACT4	ACT5	ACT6	ACT7	ACT8	ACT9	ACT10	ACT11	ACT12
ACT1	X		X	X	X							
ACT2		X						X	X			
ACT3			X			X						
ACT4				X								
ACT5					X							
ACT6						X						
ACT7							X					
ACT8								X				
ACT9					X				X			
ACT10										X		
ACT11											X	
ACT12												X

Figure 5 is a partial listing of the interactive user dialogue for defining activity ACT3. ACT3 starts at node 1 and ends at node 3. The activity type is fixed, the duration is 100, the time to failure follows an exponential distribution with a mean of 120, the time to repair the activity follows the normal distribution with a mean of 10 and a standard deviation of 2. Activity ACT6 is dependent on ACT3.

Figure 6 is a partial listing of the GPSS main program. Note that the code consists of a number of SPLIT, TRANSFER and ASSEMBLE blocks that define the network. The TRANSFER blocks route the transactions to the appropriate fixed or variable activity operation subroutines.

```
    Name for GPSS Program              : EXAMP.1

1. Number of activities                : 12
2.  Activity attributes:
    Activity name                      : $ACT3
    Activity type (fixed/variable)     : FIXED
    Duration distribution type         : CONSTANT
             mean time                 : 100

    Starting node number               : 1
    Ending   node number               : 2
    MTTF     distribution type         : EXPONENTIAL
             mean time                 : 120

    MTTR     distribution type         : NORMAL
             mean time                 : 10
             standard deviation        : 2
    Number of dependent activities     : 1

    Name of dependent activity 1:$ACT6
```

Figure 5. Partial listing of interactive user dialogue

```
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      ASSEMBLE      2
2003          TRANSFER      ,A4
2004 EV3      ASSEMBLE      2
2005          TRANSFER      ,A5
2006 EV4      ASSEMBLE      2
2007          LOGIC S       SWITCH_END1
2008 EEV4     ASSEMBLE      2
2009          TRANSFER      ,A12
2010 EV5      ASSEMBLE      1
2011          LOGIC S       SWITCH_END2
2012 EEV5     ASSEMBLE      2
2013          TRANSFER      ,END1
2014 EV6      ADVANCE
2015          TRANSFER      ,A7
2016 EV7      ADVANCE
2017          TRANSFER      ,A9
2018 EV8      ADVANCE
2019          TRANSFER      ,A11
2020 A1       ASSIGN        2,$ACT1
2021          SPLIT         1,A2
2022          ASSIGN         3,1
2024          LOGIC R       SWITCH_END1
2025          TRANSFER      SBR,VENT_B,RTRN2
2026          TRANSFER      ,EEV4
2027 A2       ASSIGN        2,$ACT2
2028
              TRANSFER      ,-
2082 A10       ASSIGN        2,$ACT10
2083          ASSIGN        3,10
2085          TRANSFER      SBR,VENT_A,RTRN2
2086          TRANSFER      ,EV8
2088 A11       ASSIGN        2,$ACT11
2089          ASSIGN        3,11
2091          TRANSFER      SBR,VENT_A,RTRN2
2092          TRANSFER      ,EV4

2094 END1     TABULATE      SYSTIME
2095 SYSTIME  TABLE         MP$SYSTIME,0,50,50
2096          LOGIC S       SWITCH_MORE
2097          TERMINATE     1
```

Figure 6. Partial GPSS listing of main program

CONCLUSIONS

The ANPS system is currently in limited operation on an IBM PC microcomputer. A number of relatively small network problems have been solved using the system. Given the success in modeling these small networks, it appears that the ANPS system can readily model the two large Saturn V prelaunch models by Synder (1967) and Schroer (1969). Based on this initial testing and evaluation, the following comments can be made:

- The interactive user dialogue provides for a formal and structured procedure for acquiring information on the network being modeled.
- The interactive user dialogue expedites the definition of the problem specification and assures a complete and detailed definition of the problem specification.
- The automatic code generator results in structured simulation code that is easy to read, trace and modify.
- The overall clarity of the simulation code is greatly improved.
- The ANPS system is ideal for rapid prototyping and can produce simulation code that is syntax error free.
- The ANPS system reduces the knowledge level required by the modeler of the simulation language.

The ANPS system also has several disadvantages. These disadvantages include:

- The system is domain specific and limited by the robustness of its library of macros.
- The GPSS code generated by ANPS probably is longer, and consequently requires more memory and takes longer to execute, than a nonstructured equivalent program.

A second version of ANPS is currently under development on an Apple Mac II using HyperCard. This version uses an interactive graphical interface rather than the interactive user dialogue. With this version it will be possible to compare the different interface approaches to defining the problem specification, the use of Turbo Prolog versus HyperCard, and the PC and Mac II platforms.

ACKNOWLEDGEMENTS

This research was funded in part by grant NAG8-641 from the NASA Marshall Space Flight Center and contract ADECA-UAH-9001 from the Science, Technology, and Energy Division of the Alabama Department of Economic and Community Affairs.

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. 1969. "Saturn V Prelaunch Systems Simulation Model for a Launch Opportunity Containing Multiple Launch Windows," 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.

Synder, J. E. R. Bennich and Y. H. Lindsey. 1967. "Implementation of Advanced Simulation Techniques for Predicting the Saturn V Launch Vehicle System Behavior," AIAA 5th Aerospace Sciences Meeting, Paper 67-205, New York, January 1967.

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