Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

COMBINING SOFTWARE ENGINEERING PRINCIPLES WITH DISCRETE EVENT SIMULATION

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

Shou X. Zhang
Northwestern Polytechnical University
Xian, Shaanxi
Peoples Republic of China

ABSTRACT

This paper presents an approach for using the concepts of software engineering to improve the simulation modeling environment. This approach is demonstrated by discussing the Automatic Manufacturing Programming System with a graphical user interface.

1.0 INTRODUCTION

The concepts of software engineering offer an approach to minimizing software development problems and to improving the overall simulation modeling environment. Software engineering encompasses the entire life cycle process by which a program is conceptualized, structured, programmed, verified, validated, and maintained. The goal of software engineering is to develop quality code, on time, and within budget. To meet this goal requires a variety of programming tools such as a good language with a library of reusable modules, a flexible editor, and a potent debugger.

2.0 MODELING LIFE CYCLE

Figure 1 outlines the phases of the model life cycle, or the model development (Balci 1986 and Nance 1988). Basically, the modeling process is iterative rather than sequential as indicated in Figure 1. That is, the modeler goes back and forth between the various phases during the modeling process.

Rapid prototyping is a technique used in software engineering for capturing system requirements early in the modeling life cycle so that these requirements can be evaluated, tested, verified and validated early in the process before starting the actual coding. The end result of rapid prototyping is the potential for large increases in productivity.

An element of rapid prototyping is the automatic conversion of the communicative model into executable code. Automatic Programming (AP) is defined as the automation of some aspects of the computer programming process (Barr 1982). This automation is accomplished by developing another program, an automatic programming system, that raises the level of specifying computer program instructions. In other words, an AP system helps programmers write programs. AP systems improve the overall environment for defining and writing programs (Brazier and Shannon 1987). Consequently, there should be a reduction in the amount of detail that the programmer needs to know.

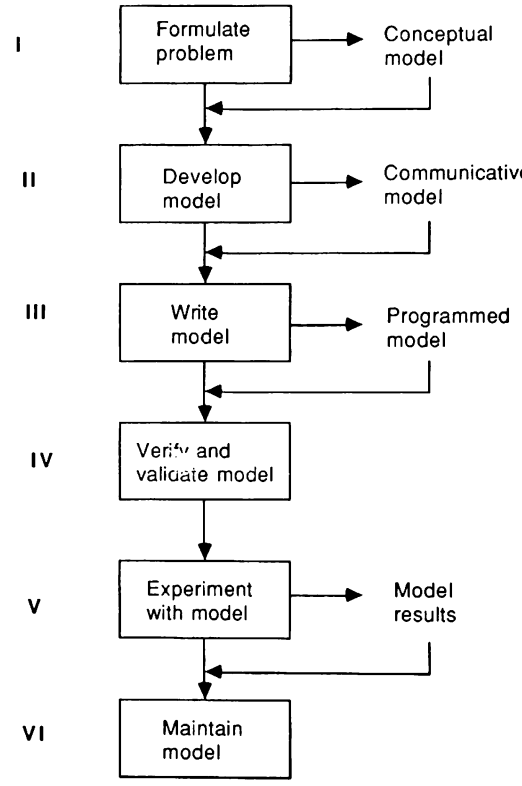

Figure 1. Phases in the modeling life cycle

Figure 2 shows the overlaying of automatic programming onto the modeling life cycle in Figure 1. Phase II , model development, has been replaced by a user interface program that assists the modeler in defining the problem specification. Three approaches are commonly used by the user interface in defining the problem specification. These approaches are an interactive dialogue interface, a graphical interface, and a natural language interface.

Also, Phase III, write model, has been replaced by an automatic code generation program. Many code generators take advantage of the concept of code reusability and have a library of predefined software modules or macros. These macros are written in the target simulation code and are merged within the simulation code.

3.0 RESEARCH OBJECTIVE

The focus of this paper is on using the concepts of software engineering to improve the simulation modeling environment for modeling manufacturing systems. Of special interest is to apply an element of rapid prototyping, or automatic programming, to assist the modeler define the problem specification as defined by Phase II in Figure 2. Then, once the problem specification has been defined, an automatic code generator is used to write the simulation code. This step is defined by Phase III in Figure 2.

4.0 AUTOMATIC MANUFACTURING PROGRAMMING SYSTEM

The Automatic Manufacturing Programming System (AMPS) is a software engineering tool for rapidly prototyping selected phases of the simulation process for domain specific manufacturing systems. The AMPS system consists of the following elements:

- Set of generics manufacturing modules written in GPSS/PC (Minuteman 1986)
- An interface program for extracting the problem from the user and for creating a problem specification file
- An automatic code generator program for creating the code in the target simulation language GPSS/PC

A thorough review was made of a number of manufacturing systems. This review resulted in a set of functions that was common to manufacturing systems. These common functions are:

- Tasking
- Assembly
- Inspection
- Inventory transfer
- Manufacturing

The three common user interface programs for extracting the problem from the user and for creating the problem specification file are: interactive dialogue interface, interactive graphical interface, and natural language interface. The AMPS system was initially developed using the interactive dialogue interface (Schroer and Tseng 1988). The AMPS system takes the user through a series of questions. Based on the responses by the user, the system follows various branches in defining the problem specification. Initially, the AMPS system was written in Lisp on a Symbolics 3620 workstation. Since the majority of the end users did not have a Symbolics, a second system was written in Pascal for the IBM PC.

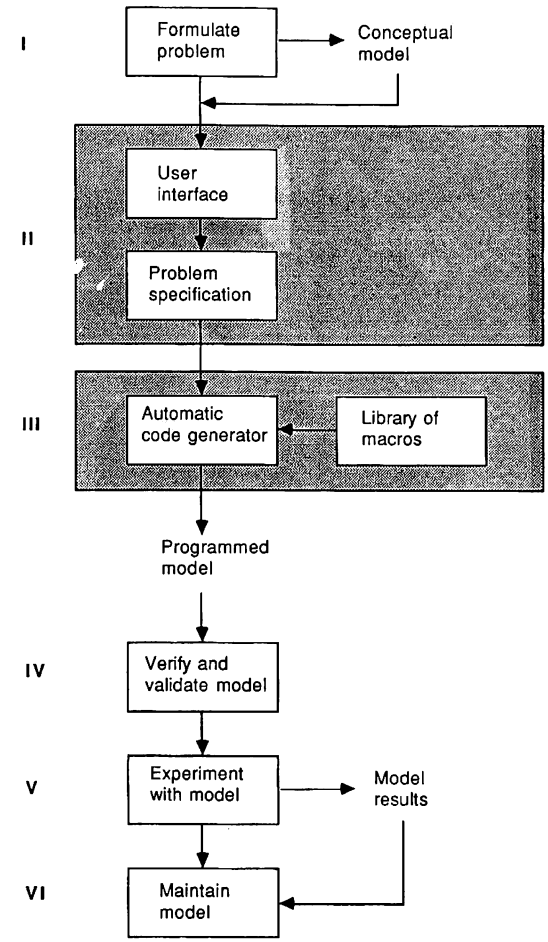

Figure 2. Software engineering imbedded in the modeling life cycle

The AMPS/Graphics uses a graphical interface. The user now defines the problem specification by creating a graphical process flow chart on the workstation. This paper focuses on the graphical interface program. Special emphasis is on the structure of AMPS/Graphics, the system operation, and a comparison with the original interactive dialogue interface.

There were several reasons for initially selecting the interactive user interface approach. First, the overall coding was the easiest of the three approaches. Second, changes could be readily made to the program. And third, since the overall approach was still being finalized, an initial prototype could be delivered more quickly.

Figure 3 gives a comparison of the three approaches. Of special interest is the amount of time to develop each system. Both of the versions developed on the Symbolics were written by the same individual with the graphics versions written last. Notice that the interactive dialogue version required six man-months while the graphics version required fifteen man-months. The Pascal version required three months and was written by another person.

Parameter	Interactive dialogue interface		Interactive graphical interface
	AMPS	AMPS/PC	AMPS/Graphics
Development platform	Symbolics 3620	IBM PC	Symbolics 3620
Programming language	Lisp	Turbo Pascal	Lisp
Lines of code	1500	1900	3500
Time to develop program (man-months)	6	3	15

Figure 3. Comparison of AMPS versions

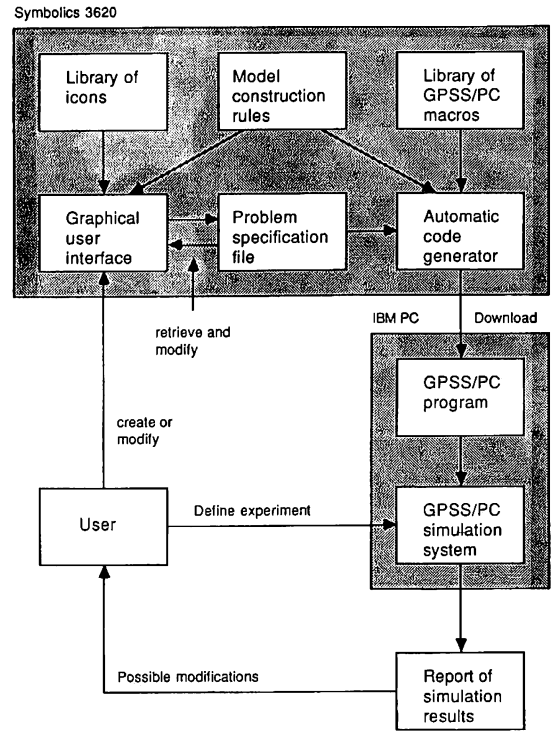

Figure 4. AMPS/Graphics system overview

5.0 AMPS/GRAPHICS OVERVIEW

An overview of the AMPS/Graphics system is given in Figure 4. The user sits at a Symbolics 3620 workstation to create or modify the model. The output of the graphical user interface program is the problem specification file. The automatic code generator program combines the specification file with the selected GPSS/PC macros and writes the simulation program. The program is then downloaded to the IBM PC and executed by the GPSS/PC system. To modify the program, the user recalls the problem specification file and the cycle repeated.

6.0 AMPS/GRAPHICS DESCRIPTION

The tree structure of the AMPS commands is given in Figure 5. The system consists of five menus: Main, Model, Layout, Specification and GPSS. In summary, the Main Menu contains the master control commands. The Model Menu contains the commands for creating, editing, saving, and reading models. The Layout Menu contains the commands for constructing the model. The Specification Menu includes the commands for defining the model parameters. The GPSS Menu contains the commands for writing the simulation code.

Figure 6 is a list of the icons available in AMPS. These icons serve as the construction blocks in defining a manufacturing system. To define a manufacturing system the user selects these icons and develops a process flow showing the various stations and the flow between the stations. Figure 7 gives all the feasible connections between the icons. For example, it is not feasible to connect an inspection station to a manufacturing cell.

The function and connection rules for each of the icons are documented within the system. The user can click on an icon to learn the function and the rules of the icon. All the connection rules are implemented in the system as construction rules of the models. As the user creates a model, the AMPS checks the partially completed model immediately for possible local violations of the rules. For example, Figure 8 shows the rules for an assembly station.

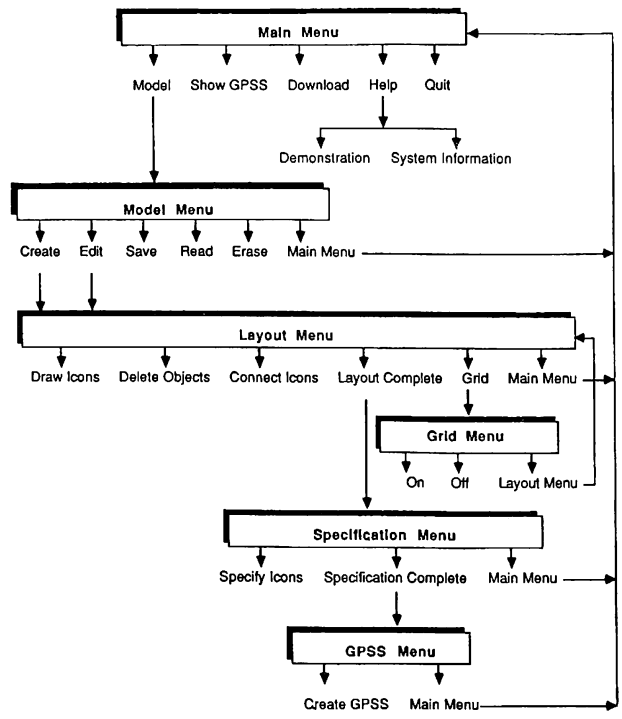

Figure 5. AMPS/Graphics commands

Icon	Function
	Assembly station
	Starting point of an assembly line
	Demand stock point of pull inventory system
	Ending stock point of final product
	Inspection station
	Manufacturing cell
	Stock point for part ordered from outside
	Stock point for push inventory system
	Supply stock point of pull inventory system
	Task station

Figure 6. Library of AMPS/Graphics Icons

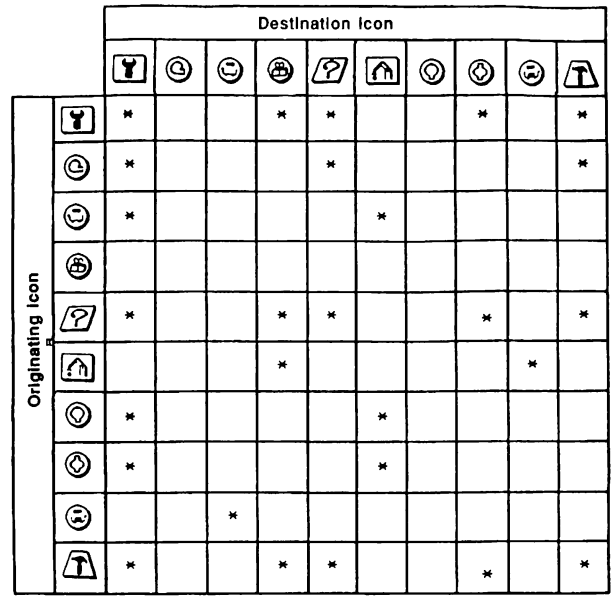

Figure 7. Valid AMPS/Graphics Icon connections

When the process flow has been completely drawn, the AMPS/Graphics will check the completeness of the structure. After the layout has passed the check for completeness, the user enters the parameters of the manufacturing system. The user then clicks on each icon to input the specification. A parameter menu will pop up on the screen. Figure 9 shows the parameter menu of an inspection station. The user can move the cursor to each field to enter the data. The system then performs additional checking. For example, the AMPS will check whether the data are the right types for the fields. The AMPS will make certain that an initial inventory level is not larger than the capacity.

7.0 SAMPLE PROBLEM

Figure 10 is an example of a simple manufacturing system formulated using the AMPS/Graphics system. The manufacturing system consists of an assembly line, MAIN and two assembly stations, STA1 and STA2. The assembly line produces part A. Station STA1 assembles part C to the incoming part and passes it to station STA2. Station STA2 then assembles part B to the incoming part from station STA1 and produces part A. Part C is supplied through a pull inventory control system from manufacturing cell MC. A part C is made of parts D and E at the manufacturing cell MC. Parts B, D, and E are supplied from outside sources.

Parts arriving at the assembly line follow the exponential distribution with a mean of 100. The assemble time of each of the two stations is a constant 100. Station STA1 requires one part C and station STA2 requires one part B for an assembly. The stock point to hold the final product, part A, has a capacity of 1000 units.

Part C is used at station STA1 and is manufactured at manufacturing cell MC. A pull inventory system controls the production and shipment of part C, which is represented by a pair of supply and demand stock points. A vehicle WGIG is used to move the carts between the stock points. The time to move the carts is 10. Each cart has a capacity of 4 parts C. Initially there is a cart of parts C at each of the supply and demand stock points. Parts B, D, and E are supplied from outside sources. Initially there are 1000 units for each part type.

Manufacturing cell MC makes part C. One part D and two parts E are used to make one part C. The manufacturing time is 100 and there is no setup time.

The model is created by selecting the Model command from the Main Menu and the Create command from the Model Menu (See Figure 5). The actual layout of the model is created by using the commands Draw Icons and Connect Icons in the Layout Menu.

After the model has been completely drawn, the Layout Complete option is selected to start specifying the model parameters. Figure 11 shows a portion of the model parameters. To specify an icon the user simply clicks on the icon when the AMPS is in the Specification Menu.

Both the layout and the parameters can be saved for future use through the Save command in the Model Menu. At the completion of the problem specification, the user selects the Specification Complete command to end the model specification. The system then leads the user to the GPSS Menu command to create the corresponding simulation code in the target language GPSS/PC.

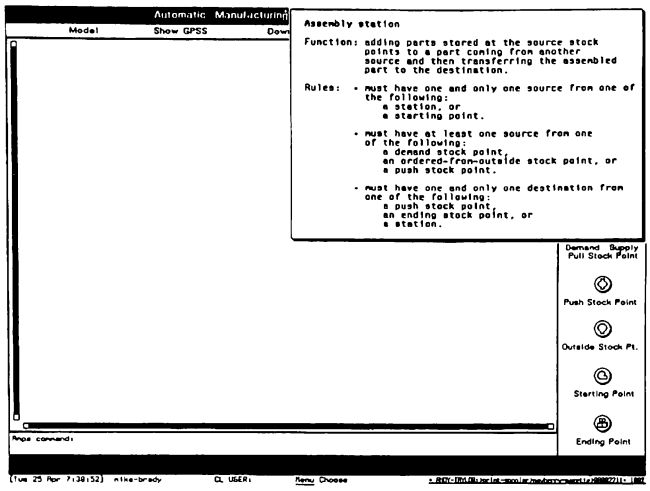

Figure 8. Assembly station rules

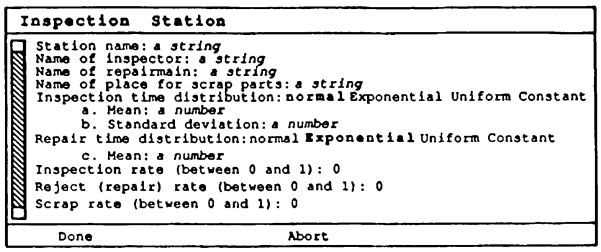

Figure 9. Inspection station parameters

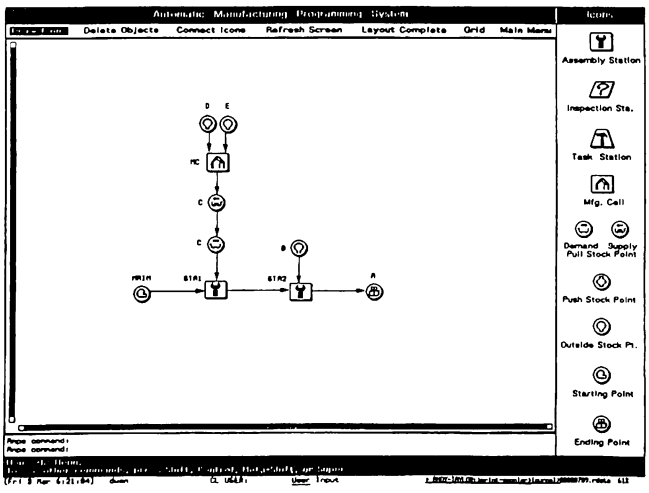

Figure 10. Sample manufacturing system

```
Starting Point of Line
   Name of line: MAIN
   Interarrival time distribution: Constant
      Constant: 100


Final Product from Assembly Line
   Part name: A
   Capacity and initial inventory at the stock point:
      Maximum number of parts at stock point: 1000
      Initial number of parts at stock point: 0


Supply Stock Point
   Part name: C
     In a pull system, parts are assumed to be ordered,
     made, and shipped by carts. Two stockpoints: supply
     and demand are created.
   Capacity and initial inventory at the stock point:
      Current cart capacity (number of parts per cart): 4
      Initial number of carts at demand stock point: 1
      Initial number of carts at supply stock point: 1
   Vehicle used to move carts between stock points: wgig
   Moving time distribution: Constant
      Constant: 10


Ordered from outside
   Part name: D
   Capacity and initial inventory at the stock point:
      Maximum number of parts at stock point: 1000
      Initial number of parts at stock point: 1000
   Will Part D be replenished during the simulation? No
```

Figure 11. Partial parameter input

8.0 CONCLUSIONS

A system such as AMPS/Graphics offers a number of advantages for improving the simulation modeling environment. One advantage is rapid prototyping. Once the necessary library of GPSS macros has been written, the system permits the user to rapidly construct a model. As a result, the AMPS/Graphics produces executable simulation code that is syntax error free.

A second advantage is improved clarity of the simulation code. The GPSS code generated by the system is structured simulation code that is easy to read, trace, and modify.

A third advantage of AMPS/Graphics is increased productivity of the modeler. By using the system, an increase should be realized in the lines of simulation code written per hour. Several other advantages are easier maintenance of the model and a reduction in the modeler's knowledge of the simulation language.

There are also several disadvantages to a system such as AMPS/Graphics. First, and most importantly, is the system is very domain specific. Therefore, AMPS/Graphics can only model a very limited class of problems. A related disadvantage is the model robustness of the library of predefined macros. Generally skilled GPSS programmers are needed to write any new macros.

9.0 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, Alabama Department of Economic and Community Affairs.

10.0 REFERENCES

Balci, O., 1986, "Requirements for Model Development Environments," Computers and Operators Research, Vol. 13, No. 1, pp. 53-67.

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," Proceedings 1987 Winter Simulation Conference, Atlanta, GA, (December) pp. 703-708.

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

Nance, R. E., and J. D. Arther, 1988. "The Methodology Roles in the Realization of a Model Development Environment," Proceedings 1988 Winter Simulation Conference, San Diego, CA, pp. 220-225.

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

AUTHORS' BIOGRAPHIES

Bernard J. Schroer is director of the Johnson Research Center and a professor in the Department of Industrial and Systems Engineering at the University of Alabama in Huntsville. He has a PhD in industrial engineering from Oklahoma State University and is a registered professional engineer. Dr. Schroer is a member of IIE, SME/RI, SCS, AAAI, NSPE, and Sigma Xi.

Fan T. Tseng is Assistant Professor of Management Science at the University of Alabama in Huntsville. He received his PhD in Operations Research from the University of Texas at Dallas. His current research interests include simulation, expert system design, automation of manufacturing systems, and applied operations research.

Shou X. Zhang is Associate Professor in the Aircraft Manufacturing Engineering Department and deputy director of the Robotics Research Center at Northwestern Polytechnical University, Xian, Peoples Republic of China. He currently is a visiting scholar at the Johnson Research Center, University of Alabama in Huntsville.