Automatic Programming of Simulation Models

Task 3

Final Report
D.O. 34 Task 3

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

NASA/MSFC contract NAS8-36955  
Delivery Order 34

January 1990
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.0 MODELING LIFE CYCLE</td>
<td>2</td>
</tr>
<tr>
<td>3.0 SOFTWARE ENGINEERING IN THE MODELING LIFE CYCLE</td>
<td>3</td>
</tr>
<tr>
<td>3.1 Problem specification</td>
<td>5</td>
</tr>
<tr>
<td>3.1.1 Natural language interface</td>
<td>7</td>
</tr>
<tr>
<td>3.1.2 Interactive graphical interface</td>
<td>7</td>
</tr>
<tr>
<td>3.1.3 Interactive dialogue interface</td>
<td>8</td>
</tr>
<tr>
<td>3.2 Automatic simulation code generation</td>
<td>8</td>
</tr>
<tr>
<td>4.0 AUTOMATIC MANUFACTURING PROGRAMMING SYSTEM (AMPS)</td>
<td>9</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>9</td>
</tr>
<tr>
<td>4.2 AMPS System overview</td>
<td>10</td>
</tr>
<tr>
<td>4.3 Library of GPSS macros</td>
<td>10</td>
</tr>
<tr>
<td>4.4 Sample problem</td>
<td>14</td>
</tr>
<tr>
<td>5.0 AUTOMATIC NETWORK PROGRAMMING SYSTEM (ANPS)</td>
<td>16</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>16</td>
</tr>
<tr>
<td>5.2 Previous research</td>
<td>17</td>
</tr>
<tr>
<td>5.3 ANPS System overview</td>
<td>17</td>
</tr>
<tr>
<td>5.4 Library of GPSS Macros</td>
<td>18</td>
</tr>
<tr>
<td>5.5 Sample problem</td>
<td>20</td>
</tr>
<tr>
<td>6.0 DEVELOPMENTAL AP SYSTEM</td>
<td>24</td>
</tr>
<tr>
<td>6.1 AMPS/Symbolics</td>
<td>28</td>
</tr>
<tr>
<td>6.2 ANPS</td>
<td>28</td>
</tr>
<tr>
<td>6.3 AMPS/PC</td>
<td>28</td>
</tr>
<tr>
<td>6.4 AMPS/PC</td>
<td>31</td>
</tr>
</tbody>
</table>
This report contains the research results from 1988 research grant NAG8-641 from NASA/MSFC and a follow on 1989 contract NAS8-36955. Therefore, some of the results in this report were documented in the final report, Automatic Programming of Simulation Models, UAH Report 725, September 1988.
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.

The focus of this research project is on using the concepts of software engineering to improve the simulation modeling environment. Of special interest is to apply an element of rapid prototyping, or automatic programming, to assist the modeler define the problem specification. Then, once the problem specification has been defined, an automatic code generator is used to write the simulation code.

The following two domains were selected for evaluating the concepts of software engineering for discrete event simulation: manufacturing domain and a spacecraft countdown network sequence.

The specific tasks for this follow-on contract were to:
1. Define the software requirements for a graphical user interface to the Automatic Manufacturing Programming System (AMPS) system.
2. Develop a graphical user interface for AMPS.
3. Compare the AMPS graphical interface with the AMPS interactive user interface.
2.0 MODELING LIFE CYCLE

There has been considerable interest in improving the process for developing simulation models. One area of interest is the development of simulation support environments. Henriksen (1983) suggests a simulation software development environment composed of a set of integrated software tools. Standridge (1983) proposes the integration of software tools and database management techniques on each stage of the simulation model development process. Pidd (1984) also outlines a simulation support environment concept for handling one simulation problem at a time.


More recently, the Semiconductor Manufacturing Technology Initiative (SEMATECH) is developing a coherent modeling environment called CHIPS (Coherent Intergrated Planning System). CHIPS consists of five major modules: process flow analysis module, queueing network analysis module, system simulation module language, and a cost analysis module (Phillips, et al 1989). SEMATECH is a cooperative project between industry and government with the goal to recover the world leadership in semiconductor manufacturing.
Figure 1 outlines the phases of the model life cycle (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.

Figure 1 can be considered as the traditional approach to simulation modeling (Balci 1986 and Nance et al 1988). The same process also applies to general modeling problems. The process consists of six stages described on the left side of the figure. On the right side, different types of models generated at different stages through the process are listed. For example, a conceptual model of a manufacturing system may be the understanding of the system by the modeler and in the mind of the modeler. A communicative model of the manufacturing system may be a graphic representation of the system in the form of a block diagram, flowchart, or network diagram. A GPSS simulation program of the manufacturing system is a programmed model. The model results are generated by executing the program.

3.0 SOFTWARE ENGINEERING IN THE MODELING LIFE CYCLE

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
Figure 1. Phases in the modeling life cycle
program, an automation programming system, that raises the level of satisfying 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.

To write simulation programs automatically, two phases in the simulation modeling process are usually automated. The first phase is the automation of the process of specifying the problem. The second phase, and the more difficult phase, is the automatic generation of executable code in the target simulation language.

3.1 Problem Specification

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.

The automatic problem specification can be considered 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).

Three approaches for assisting the user in defining the simulation model or problem specification are:

- Natural language interface
- Interactive graphical interface
- Interactive dialogue interface
Figure 2. Software engineering imbedded in the modeling life cycle
There is a fourth approach for assisting in the definition of problem specification, which is the use of a high-level specification language. This approach is less domain specific. However, the use of a high-level specification language requires the user to learn another language in order to define a problem.

3.1.1 Natural language interface

The Natural Language Interface (NLI) allows the user to specify the problem in free text format to the computer via a keyboard. The NLI then attempts to parse the text and automatically generate the simulation code in the target language. Most NLI's communicate interactively with the user to identify missing information and possible inconsistencies. The Natural Language Programming for Queueing Simulations (NLPQ) (Heidorn 1974) and the Electronic Manufacturing Simulation System (EMSS) (Ford and Schroer 1987) are two examples of a NLI.

3.1.2 Interactive graphical interface

The second approach to assist the user in specifying the problem is an Interactive Graphical Interface (IGI), which is less difficult than the NLI. An IGI consists of a menu of icons that are mouse selectable by the user in constructing a graphical representation of the system being simulated. Once the system has been constructed, the user inputs the attributes corresponding to the icons through the keyboard.

An example of an IGI is by Khosnevis and Chen (1986) who developed an object-oriented approach for graphically modeling a system. This 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.

3.1.3 Interactive dialogue interface

The third approach to assist the user in defining the problem specification is the Interactive Dialogue Interface (IDI). An IDI consists of a series of questions that are asked the user. Among the three approaches for defining the problem specification, the interactive dialogue interface is the one most commonly used by developers.

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 System (AGVS). Murray and Sheppard (1988) have developed a Knowledge Based Model Construction (KBMC) system to automate model definition and code generation. These last three systems generate SIMAN code.

3.2 Automatic simulation code generation

In Figure 1, Phase III, write model, has been replaced in Figure 2 by an automatic code generation program. Basically, two approaches exist for taking the internal problem specification and then automatically generating executable code in the target simulation language. The first approach is to generate simulation code directly from the internal representation of the problem specification.

A 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 specialized 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.

4.0 AUTOMATIC MANUFACTURING PROGRAMMING SYSTEM (AMPS)

4.1 Introduction

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:

- A set of generic 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

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 subassembly lines.
- Inventory of parts being moved between the manufacturing cells and subassembly lines.
4.2 **AMPS System Overview**

Figure 3 is an overview of the AMPS system operation. Once the user has scoped the problem domain, the user sits at a 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 user then adds the experimental frame, such as the run statements, and the GPSS/PC simulation program is executed. To change the GPSS/PC model, the user 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.

4.3 **Library of GPSS Macros**

In analyzing most manufacturing systems at the macro level, the following function 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
User

User interface program

Problem specification

Automatic code generator program

Library of GPSS/PC macros

User defines experiment

GPSS/PC simulation program

GPSS/PC simulation system

Simulation model results

Possible problem modifications

Figure 3. AMPS system overview
These five functions represent the current domain of manufacturing functions within the AMPS system. Once the manufacturing functions have been defined, GPSS subroutines are written for the functions (see Appendix A). 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
- Manufacturing
- Inventory transfer
- Inspection
- Task

In a recent article on SEMATECH (Phillips, et al 1989), researchers have identified ten machine modules for the semiconductor manufacturing domain. Furthermore, the SEMATECH group has indicated that possibly no more than 16-20 generic machine modules may be required to completely represent the semiconductor manufacturing environment.

Figure 4 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 4, station STA1 may assemble two part C's and three part D's to the incoming part A resulting in Part B.

The manufacturing cell makes a cart of specified parts when an order is received. The cell can make multiple part types. For example, in Figure 4, cell MCI may make one part A from two part C's and three part D's.
Figure 4. GPSS manufacturing macros
and one part B from one part D. The task station performs an operation on a part. For example, in Figure 4 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.

4.4 Sample Problem

Figure 5 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, located throughout the manufacturing system.

The GPSS program for the manufacturing system in Figure 5 that was generated by AMPS consists of 344 blocks, of which 110 blocks are for the five macros, 25 blocks are for the main program and 209 blocks are matrix savevalues for defining the system attributes.
Figure 5. Manufacturing system
5.0 AUTOMATIC NETWORK PROGRAMMING SYSTEM (ANPS)

5.1 Introduction

Large simulation projects have been undertaken for the space program. One of the projects involves simulating the countdown sequence prior to spacecraft liftoff. A countdown has a number of constraints. 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 propellants. The handling of the cryogenic propellants 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 Automatic Network Programming System (ANPS) is a tool 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 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.
5.2 Previous Research

Snyder et al. (1967) have developed a simulation model of the Saturn V prelaunch activities beginning at T-24 hours and continuing through T-O 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 additions 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 Snyder 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.

5.3 ANPS System Overview

The three AP elements in ANPS are an Interactive Dialogue Interface (IDI), a library of software modules, and an automatic simulation code generator.
The actual operation of ANPS is similar to AMPS (see Figure 3). 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. The specification file is used by the code generator program to create the simulation program in the target simulation language GPSS/PC.

5.4 Library of GPSS Macros

Since the ANPS system is domain specific to prelaunch countdown sequences, the number of needed software modules is minimal. Consequently, ANPS consists of the following four GPSS modules (see Appendix B):

- 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 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 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.

The ANPS macros impose 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.

No two activities can start from the same node and terminate on
the same node.

5.5 Sample Problem

Figure 6 is a time line for a simplified prelaunch countdown
sequence consisting of 16 fixed activities and two continuous activities.
Figure 7 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 acti-
vity 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 I contains the time attributes for the activities in the pre-
launch countdown. These attributes include activity duration, activity time
to failures, and activity time to repairs. Note that activities ACT1 and
Figure 6. Prelaunch countdown for sample problem
Figure 7. Network representation of prelaunch sequence
<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration (hours)</th>
<th>Failure time (hours)</th>
<th>Repair time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continuous</td>
<td>E(33)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>2</td>
<td>Continuous</td>
<td>E(33)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>E(12)</td>
<td>N(30,3)</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>E(12)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>E(12)</td>
<td>N(45,5)</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>E(12)</td>
<td>N(45,5)</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>E(12)</td>
<td>N(45,5)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>E(12)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>E(12)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>E(12)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>E(12)</td>
<td>N(45,5)</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>E(12)</td>
<td>N(30,3)</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>E(12)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>E(12)</td>
<td>N(60,9)</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>E(12)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>E(12)</td>
<td>N(120,2)</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>E(12)</td>
<td>N(60,6)</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>E(12)</td>
<td>N(45,5)</td>
</tr>
<tr>
<td>19</td>
<td>Dummy</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>Dummy</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>Dummy</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>Dummy</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>Dummy</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table II. Operational dependencies between activities

<table>
<thead>
<tr>
<th>Activity</th>
<th>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>x</td>
</tr>
<tr>
<td>5</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>x</td>
</tr>
<tr>
<td>8</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
</tr>
<tr>
<td>11</td>
<td>x</td>
</tr>
<tr>
<td>12</td>
<td>x</td>
</tr>
<tr>
<td>13</td>
<td>x</td>
</tr>
<tr>
<td>14</td>
<td>x</td>
</tr>
<tr>
<td>15</td>
<td>x</td>
</tr>
<tr>
<td>16</td>
<td>x</td>
</tr>
<tr>
<td>17</td>
<td>x</td>
</tr>
<tr>
<td>18</td>
<td>x</td>
</tr>
<tr>
<td>19</td>
<td>x</td>
</tr>
<tr>
<td>20</td>
<td>x</td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>
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.

Table II 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.

Figure 8 gives the distribution of time to complete the prelaunch sequence in Figure 6. 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 six hour window is given in Figure 9. The LVA for a two hour window is 0.015 and increases to 0.596 for a six hour window.

6.0 DEVELOPMENTAL AP SYSTEMS

Table III contains a comparison of the six platforms that have been developed for the AMPS and ANPS systems. Two programmers were used to develop the systems. Programmer A was Mr. W.S. Dwan who was a graduate student in computer science at the University of Alabama in Huntsville (UAH). Mr. Dwan was experienced in LISP on a Symbolics workstation. Programmer B was Mr. S.X. Zhang who was a visiting scholar at UAH from Northwestern Polytechnical University in Xian, China.
Figure 8. Time to complete prelaunch countdown
Figure 9. Launch vehicle availability
### Table III  Comparison of platforms

<table>
<thead>
<tr>
<th>System</th>
<th>User interface</th>
<th>Order of development</th>
<th>Programmer</th>
<th>Platform</th>
<th>Language</th>
<th>Target language</th>
<th>Man-months</th>
<th>Lines of code</th>
<th>Lines of code per month</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMPS</td>
<td>IDI</td>
<td>1</td>
<td>A</td>
<td>Symbolics</td>
<td>Lisp</td>
<td>GPSS/PC</td>
<td>6</td>
<td>1,500</td>
<td>250</td>
</tr>
<tr>
<td>ANPS/PC</td>
<td>IDI</td>
<td>2</td>
<td>B</td>
<td>IBM/PC</td>
<td>Prolog</td>
<td>GPSS/PC</td>
<td>4</td>
<td>1,300</td>
<td>325</td>
</tr>
<tr>
<td>AMPS/PC</td>
<td>IDI</td>
<td>3</td>
<td>B</td>
<td>IBM/PC</td>
<td>Pascal</td>
<td>GPSS/PC</td>
<td>3</td>
<td>1,900</td>
<td>633</td>
</tr>
<tr>
<td>AMPS/PC</td>
<td>IDI</td>
<td>4</td>
<td>B</td>
<td>IBM/PC</td>
<td>C</td>
<td>GPSS/PC</td>
<td>4</td>
<td>1,300</td>
<td>325</td>
</tr>
<tr>
<td>AMPS/Graphics</td>
<td>IGI</td>
<td>5</td>
<td>A</td>
<td>Symbolics</td>
<td>Lisp</td>
<td>GPSS/PC</td>
<td>15</td>
<td>3,500</td>
<td>233</td>
</tr>
<tr>
<td>AMPS/PC</td>
<td>IDI</td>
<td>6</td>
<td>B</td>
<td>IBM/PC</td>
<td>C</td>
<td>SIMAN/PC</td>
<td>3</td>
<td>1,600</td>
<td>533</td>
</tr>
</tbody>
</table>

**Notes**
1. IDI - Interactive Dialogue Interface
2. IGI - Interactive Graphical Interface
3. AMPS - Automatic Manufacturing Programming System
4. ANPS - Automatic Network Programming System
6.1 AMPS/Symbolics

The AMPS system was initially developed for the Symbolics 3620 workstation and used the Interactive Dialogue Interface (IDI). Figure 10 is a portion of a typical IDI dialogue. The AMPS system was written in LISP by programmer A in six man months. The system consisted of 1,500 lines of LISP code. The code production was 250 lines per month.

A detailed description and operation of the AMPS system is given in UAH Report 720, Automated Manufacturing Programming System User’s Manual, September 1988. The system has been submitted to NASA COSMIC (reference #28367). The AMPS/Symbolics system was also ported to the TI Explorer workstation.

6.2 ANPS

ANPS was the second system developed and used with the IDI dialogue. Figure 11 is a portion of a typical IDI dialogue. This system was developed by programmer B using Turbo Prolog on an IBM/PC. The system consisted of 1,300 lines of code. The code production was 325 lines per month.


6.3 AMPS/PC

This version of AMP was developed in Turbo Pascal on an IBM/PC and uses an IDI dialogue. The system was developed by programmer B using Turbo Pascal (Borland 1987) on an IBM PC. The system consists of 1,900 lines of code. The code production was 633 lines per month.
How many types of final products to be made in the manufacturing system: 2
   Name of the product 1: A
   Name of the product 2: B

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

* Specification of product A

Type of the facility used to produce product A at the final stage: Assembly line

Name of the line to produce product A: MAIN
Number of stations in line MAIN: 2
Capacity and initial inventory at the stock points:
   Maximum number of parts at stock point: 2000
   Initial number of parts at stock point: 0

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

* Description of line MAIN

Input process (Interarrival time of orders):
   Time:
   Distribution: Exponential
   Mean: 100

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

station 1
   (1) Station id: 1
   (2) Type of station: Assembly station
   (3) Station name: ONE
   (4) Part required:
       Number of part types required: 2
       Name of part: C
       Number of part: 1
       Name of part: D
       Number of part: 2
   (5) Time:
       Distribution: Normal
       Mean: 100
       Standard deviation: 2

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

Figure 10. Typical IDI dialogue for AMPS/Symbolics
Name of GFSS program file : EXAMPLE1
Name of GFSS problem specification file: SPEC1

1. Number of activities : 7

2. Activity attributes
   Activity name : $ACT1
   Activity type (fixed/variable) : FIXED
   Duration distribution type : CONSTANT
   mean time : 20

   Starting node number : 1
   Ending node number : 5
   MTTF distribution type : CONSTANT
   mean time : 100
   MTTF distribution type : CONSTANT
   mean time : 0

   Number of dependent activities : 0

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

---

Figure 11. Typical IDI dialogue for ANPS

6.4 AMPS/PC

This version of AMP is identical to the AMPS/PC in Section 6.3. The only difference is this version is written in Turbo C (Borland 1988) for the IBM/PC. The system consists of 1,300 lines of code. The code production was 325 lines per month. This system has not been submitted to NASA COSMIC.

6.5 AMPS/Graphics

This version of AMPS was developed for the Symbolics 3620 workstation and uses the Interactive Graphical Interface (IGI). The system was written in LISP by programmer A in fifteen months. The system consists of 3,500 lines of code. The code production was 233 lines per month.

The AMP/Graphics system is documented in UAH Report 788, *Automatic Manufacturing Programming/Graphics*, August 1989. The system is being submitted to NASA COSMIC. Since the AMPS/Graphics has been developed under the followon contract, a more detailed discussion of the system follows.

6.5.1 AMPS/Graphics Overview

An overview of the AMPS/Graphics system is given in Figure 12. The user sits at a Symbolics 3620 workstation to create or modify the model. The output of the Interactive Graphical Interface (IGI) program is the
Figure 12. AMPS/Graphics system overview
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.

The tree structure of the AMPS commands is given in Figure 13. 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 14 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 15 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 these 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 16 shows the rules for an assembly station.
Figure 13. AMPS/Graphics commands
<table>
<thead>
<tr>
<th>Icons</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="Assembly station" /></td>
<td>Assembly station</td>
</tr>
<tr>
<td><img src="icon" alt="Starting point of an assembly line" /></td>
<td>Starting point of an assembly line</td>
</tr>
<tr>
<td><img src="icon" alt="Demand stock point of pull inventory system" /></td>
<td>Demand stock point of pull inventory system</td>
</tr>
<tr>
<td><img src="icon" alt="Ending stock point for final product" /></td>
<td>Ending stock point for final product</td>
</tr>
<tr>
<td><img src="icon" alt="Inspection station" /></td>
<td>Inspection station</td>
</tr>
<tr>
<td><img src="icon" alt="Manufacturing cell" /></td>
<td>Manufacturing cell</td>
</tr>
<tr>
<td><img src="icon" alt="Stock point for part ordered from outside" /></td>
<td>Stock point for part ordered from outside</td>
</tr>
<tr>
<td><img src="icon" alt="stock point for Push inventory system" /></td>
<td>stock point for Push inventory system</td>
</tr>
<tr>
<td><img src="icon" alt="Supply stock point of pull inventory system" /></td>
<td>Supply stock point of pull inventory system</td>
</tr>
<tr>
<td><img src="icon" alt="Task station" /></td>
<td>Task station</td>
</tr>
</tbody>
</table>

Figure 14. Library of AMPS/Graphics icons
Figure 15. Valid AMPS/Graphics icon connections
Assembly station

Function: adding parts stored at the source stock points to a part coming from another source and then transferring the assembled part to the destination.

Rules:
1. must have one and only one source from one of the following:
   a. station, or
   b. starting point.
2. must have at least one source from one of the following:
   a. demand stock point,
   b. an ordered-from-outside stock point, or
   c. push stock point.
3. must have one and only one destination from one of the following:
   a. push stock point,
   b. an ending stock point, or
   c. a station.
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 17 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.

6.5.2 Sample Problem

Figure 18 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.
Figure 17. Typical parameter input

Figure 18. Sample manufacturing system
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 13). 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 19 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.

Appendix D contains another sample problem. Included in this appendix are layout of the manufacturing system, a listing of the input parameters, and a complete listing of the GPSS simulation model.
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

Demand 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: wqig
Moving time distribution: Constant
Constant: 10

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: wqig
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 19. Partial parameter input
6.6 AMPS/PC/SIMAN

The basic AMPS/PC system in Section 6.4 was modified to create SIMAN (Pegden 1985) rather than GPSS/PC code. This system used the identical Interactive Dialogue Interface (IDI) as the AMPS/PC system. However, the automatic code generator program was rewritten to create code in the target simulation language SIMAN. A listing of the SIMAN macros is given in Appendix C.

The system was written by programmer B in Turbo C on an IBM/PC. The system consisted of 1,600 lines of code. The code production was 533 lines per month. This system has not been documented and has not been submitted to NASA COSMIC.

7.0 SYSTEM EVALUATION

The concepts developed in AMPS have been used to model three real world problems. The first system was a Flexible Manufacturing System (FMS) at Rexham Speedring Inc., in Cullman, Alabama. The FMS consisted of 18 stations and nine alien stations. The FMS makes four different parts with each requiring 47, 31, 22 and 22 operations respectively (Schroer 1988).

The second system was a 25 station assembly line at SCI Manufacturing Inc. in Huntsville, Alabama. The line assembles a health monitoring device (Schroer 1988). The third system was a twelve station Unit Production System (UPS) at Camptown Togs, Inc. in Clanton, Alabama (Schroer and Ziemke 1989).

The following observations are made based on the above implementations:

• The problem domains were sufficiently different that the AMPS user
interface could not be used in defining the problem specification.

- The library of GPSS modules were used extensively in writing the simulation models. For the FMS model, several additional simulation modules were developed.

- By using the library of GPSS modules, the UPS model was written and validated in less than four hours as compared to forty hours without the use of the modules.

- The use of the GPSS modules caused the resulting simulation code to be structured code and well documented.

8.0 PUBLICATIONS

The following is a list of publications resulting from the research supported by NASA Grant NAG8-641 and NASA contract NAS8-36995.


9.0 CONCLUSIONS

9.1 Comparison of the AMPS/GRaphics System with the AMPS System

Both AMPS and AMPS/Graphics were written in Lisp on the Symbolics 3620 machine. The AMPS/Graphics used the Symbolics system dependent features such as the flavors (frame) window, and the graphics function. Also, AMPS/Graphics use object oriented programming concepts. The adoption of the above features greatly simplified the programming effort for such a complicated system. However, these system dependent features also make the conversion of the AMPS/Graphics to other types of machines very difficult. On the other hand, the AMPS system used very few system dependent features. Most of the statements in AMPS are Common Lisp compatible. Therefore, it is much easier to convert AMPS to other platforms. For example, the AMPS system has been successfully ported converted to a TI Explorer with only minor modifications.

The AMPS system provides an Interactive Dialogue Interface (IDI) for the user to create the model. In AMPS, the user must follow the preset logic system and answer a series of questions prompted in constructing a model. That is, the user is in a passive role. The AMPS system controls the main logic. The AMPS system allows the user to make only very limited modifications throughout the development process. Also, the user must remember the stage of the development process. Consequently, it is difficult to visualize the development process of the model in AMPS.

The AMPS/Graphics has a an Interactive Graphic Interface (IGI) through which the user builds the model mainly by icons. The user can start building the model from any part of the model. Also, the user can
always see the partially completed model on the screen. The AMPS/Graphics system allows the user to modify any part of the model throughout the development process. Consequently, it is much easier to build a model and to trace the logic by the AMPS/Graphics. Furthermore, a graphical model is also a much better communicative model than a descriptive model.

The AMPS/Graphics system provides several help features. For example, on-line documentation of each icon can be obtained by clicking a button. The documentation shows the function of each icon and the connection rules with other icons. In the construction process, if a mistake is made, the system will immediately give the appropriate error message. The AMPS system does not have these help features.

The models created by the AMPS/Graphics can be saved and then modified through the IGI interface. The corresponding simulation program will then be modified automatically. The AMPS system does not have this capability.

It is much slower to design a user friendly system such as AMPS/Graphics at the beginning of the design process because of the many factors to be considered. However, once the basic framework of the system is completed, a system such as AMPS/Graphics is much easier to modify and expand. For example, it is rather easy to add a new facility icon or to change a model construction rule. A carefully designed system should be flexible enough to add or remove a construct from the system with only minor effort. On the other hand, a system like AMPS is easier to initially design, and therefore is ideal to serve as a prototype. However, any change after the initial design requires a major modification to the system.
The AMPS/Symbolics system makes use of some advantages of the Symbolics machine. For example, the system automatically checks for some types of errors, executes much faster, and has a large amount of memory available. However, currently none of the popular commercially available discrete simulation languages, such as GPSS, is available on the Symbolics. The simulation programs must be downloaded to an IBM-PC to run the simulation. On the other hand, the AMPS/PC system is much slower than the AMPS/Symbolics. The small memory of the PC's also limits any reasonably large models to be constructed on the AMPS/PC.

9.2 **Summary**

In summary, an Automatic Programming (AP) system, such as AMPS and ANPS, offers a number of advantages for improving the simulation modeling environment. These advantages include:

- **Rapid prototyping** - Once the necessary library of simulation modules has been written, the AP system permits the user to rapidly construct a model. As a result, the AP system produces executable simulation code that is syntax error free.

- **Software correctness** - Correct simulation software requires the definition of a complete and formal set of model requirements. An AP system forces the user to completely define these requirements.
* Improved clarity of simulation code - The simulation code generated by the AP system is structured code that is easy to read, trace, and modify. An added benefit is embedded code documentation.

* Increased productivity - By using an AP system, the modeler should have an increased productivity in the lines of simulation code written per hour.

* Automatic documentation - Instead of changing program code, the user modifies the problem specification through the AP system's user interface. The AP system then rewrites a new simulation model. Therefore, the problem specification file always reflects the current configuration, or documentation, of the problem.

* Software reusability - Software reusability refers to the ability of new simulation models to use element of other models. Large collections, or libraries, of reusable program modules can be defined, making it possible to develop new models by writing only a small amount of new code. The library of GPSS modules provides the basic building blocks for the simulation model. This library is constantly being updated and expanded as the AP system is used in other domains.

* Software compatibility - Software compatibility is the ability of program modules to be interfaced with other simulation code. An AP system designed with expansion in mind and as generic as
possible will be easier to modify as additional requirements are defined.

- Extendability - Since an AP system operates in a structural environment, the overall software maintenance is less difficult. Software designed and developed using an AP system will have each data element and related processes grouped into one location, making modifications simpler.

- Reduced simulation knowledge - An advantage of an AP system is a reduction in the modeler's knowledge of the simulation language.

There are also a number of disadvantages of an AP system such as AMPS and ANPS. These disadvantages include:

- Domain specific - Most AP systems are very domain specific. Therefore, the systems can only model a very limited class of problems. To model a slightly different problem in a similar domain may require additional modules and modifications to the user interface.

- Library robustness - A related disadvantage is the robustness of the library of predefined modules. Generally skilled GPSS programmers are needed to write a new modules.
Memory and execution time - Another disadvantage is that AP systems require more memory and execute slower than a nonstructured equivalent simulation program. However, this disadvantage is not as significant as in prior years because computers are now faster and have more memory.

In comparing the IDI and IGI for the Symbolics systems, the following observations are made:

- The IGI had 3,500 lines of code versus 1,500 lines for the IDI. Interestingly, the code production was similar for both systems.
- The IGI, or object oriented approach, is preferred by the user.

10.0 ACKNOWLEDGEMENTS

In addition to the funds received from NASA/MSFC, a contract was received in 1988 from the Science, Technology, and Energy Division, Alabama Department of Economic and Community Affairs (ADECA) (Reference Contract ADECA-UH-9001) and a grant in 1989 from the Alabama Industrial Development Training (AIDT).

11.0 REFERENCES


Pegden, C. Dennis, 1985. *Introduction to SIMAN*, Systems Modeling Corp., College Station, PA.


Appendix A

GPSS macros for AMPS
GPSS Assembly station subroutine

2370 ***************************************************
2380 * ASSEMBLY STATION *
2390 ***************************************************
2400 ASM ASSIGN 3, MX$STAN(P2,1)
2410 ASSIGN 7, MX$STAN(P2,2)
2420 ASSIGN 6, MX$STIME(P2,1)
2430 ASSIGN 8,1
2440 ASSIGN 9,2
2450 QUEUE P3
2460 PAQ ASSIGN 8++,2
2470 ASSIGN 9++,2
2480 ASSIGN 5, MX$STAN(P2,P8)
2490 ASSIGN 10, MX$PART(P5,1)
2500 ASSIGN 20, MX$STAN(P2,P9)
2510 QUEUE P10
2520 TRANSFER SBR, TAKEP, RTRN2
2530 DEPART P10
2540 LOOP 7, PAQ
2550 SEIZE P3
2560 DEPART P3
2570 ADVANCE V*6
2580 RELEASE P3
2590 TRANSFER P, RTRN1,1

GPSS Task station subroutine

1980 ***************************************************
1990 * TASK STATION *
2000 ***************************************************
2010 TASK ASSIGN 3, MX$STAN(P2,1)
2020 ASSIGN 6, MX$STIME(P2,1)
2030 QUEUE P3
2040 SEIZE P3
2050 DEPART P3
2060 ADVANCE V*6
2070 RELEASE P3
2080 TRANSFER P, RTRN1,1
GPSS Inspection station subroutine

2090 ****************************
2100  * INSPECTION STATION  *
2110 ****************************
2120  INSP  ASSIGN  3, MX$STAN(P2,2)
2130  ASSIGN  4, MX$IPERC(P3,1)
2140  ASSIGN  5, MX$ITIME(P3,1)
2150  ASSIGN  6, MX$ITIME(P3,2)
2160  QUEUE  MX$STAN(P2,1)
2170  DEPART  MX$STAN(P2,1)
2180  TRANSFER  ,FN*4
2190  CHECK  QUEUE  MX$ISTA(P3,1)
2200  SEIZE  MX$ISTA(P3,1)
2210  DEPART  MX$ISTA(P3,1)
2220  ADVANCE  V*5
2230  RELEASE  MX$ISTA(P3,1)
2240  ASSIGN  4, MX$IPERC(P3,2)
2250  TRANSFER  ,FN*4
2260  REPAIR  QUEUE  MX$ISTA(P3,2)
2270  SEIZE  MX$ISTA(P3,2)
2280  DEPART  MX$ISTA(P3,2)
2290  ADVANCE  V*6
2300  RELEASE  MX$ISTA(P3,2)
2310  ASSIGN  4, MX$IPERC(P3,3)
2320  TRANSFER  ,FN*4
2330  PASS  TRANSFER  P,RTRN1,1
2340  SCRAP  QUEUE  MX$ISTA(P3,3)
2350  DEPART  MX$ISTA(P3,3)
2360  TERMINATE
GPSS Manufacturing cell subroutine
GPSS Inventory transfer subroutine
Appendix B

GPSS macros for ANPS
1360 *
1370 * ACTIVITY TIME SIMULATION GENERATOR
1380 *
1390 VENT_A SEIZE P2
1395 ASSIGN 99,MX*WORK_TIME(P3,1)
1400 ASSIGN ETIME,V*99
1405 BACK3 ASSIGN 98,MX*F_TIME(P3,1)
1410 ASSIGN MTTF,V*98
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
1530 *
1832 * CONTINUOUS ACTIVITY TIME SIMULATION GENERATOR
1834 *
1840 VENT_2 SEIZE P2
1842 ASSIGN 98,MX*F_TIME(P3,1)
1843 SAVEVALUE FTS,V*98
1845 TIME9 FVARIABLE X*FTS
1850 TIMES 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 BACK3 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
1060 * ACTIVITY FAILURE SIMULATION GENERATOR (DIRECTLY)
1070 *
1080 *
1090 FAIL ASSIGN NR_ACTS, X*ACTS
1100 ASSIGN COL, 1
1102 ASSIGN 100, MX$R_TIME(P$ROW, 1)
1105 MSAVEVALUE R$1_TIME, P$ROW, 1, V*100
1110 BACK0 TEST NE MX$ACT_NAME(P$ROW, P$COL), 0, AA
1120 GATE U MX$ACT_NAME(P$ROW, P$COL), AA
1130 SPLIT 1, AA
1140 TRANSFER , AA
1150 AB MARK DELAY
1160 FREEMPT MX$ACT_NAME(P$ROW, P$COL)
1170 ASSIGN ADELAY, MF$DELAY
1180 TEST LE PM$DELAY, 0, XACT_DELAY
1190 MSAVEVALUE R2_TIME, P$ROW, P$COL, MX$R1_TIME(P$ROW, 1)
1200 BACK1 ADVANCE MX$R2_TIME(P$ROW, P$COL)
1210 BUFFER
1220 RETURN MX$ACT_NAME(P$ROW, P$COL)
1230 TERMINATE
1240 AA ASSIGN COL+, 1
1250 LOOP NR_ACTS, BACK0
1260 TRANSFER P, RTRN1, 1

1270 *
1280 * LOGIC WHEN ACTIVITY ALREADY IS INTERRUPTED
1290 *
1300 XACT_DELAY ADVANCE
1310 MSAVEVALUE R2_TIME, P$ROW, P$COL, V$NEWDELAY
1320 NEWDELAY FVARIABLE MX$R1_TIME(P$ROW, 1) - PM$DELAY
1330 TEST L MX$R2_TIME(P$ROW, P$COL), 0, BACK1
1340 MSAVEVALUE R2_TIME, P$ROW, P$COL, 0
1350 TRANSFER , BACK1

B-3
Appendix C

SIMAN macros for AMPS
*** Assembly station model ***

ASM
ASSIGN:A(1)=M;
ASSIGN:A(2)=0;

BACKASM
ASSIGN:A(2)=A(2)+1;
BRANCH,1:
   IF,A(2).GT.A(4),OUT:
   ELSE,DOWN1;

DOWN1
BRANCH,2:
   ALWAYS,BACKASM:
   ALWAYS,DOWN2;

DOWN2
ASSIGN:M=3+2*A(2);
ASSIGN:A(5)=A(M);
ASSIGN:A(6)=A(M+1);
ASSIGN:A(11)=A(5)+3;
BRANCH,2:
   IF,P(A(11),2).EQ.2,PULL1:
   ALWAYS,PICKPT;

PICKPT
ASSIGN:M=A(5);
QUEUE,M+1;
SEIZE:PART(M),A(6);

DOWN3
ASSIGN:M=A(1);
QUEUE,M;
SEIZE:STATIONN(M);
DELAY:ED(A(3));
RELEASE:STATIONN(M):NEXT(LOOP);

PULL1
ASSIGN:A(11)=A(5)+3;
ASSIGN:A(12)=A(5)+1;
BRANCH,1:
   IF,NR(A(12)).GE.P(A(11),4),PICKOAR:
   ELSE,OUT;

PICKCAR
ASSIGN:M=P(A(11),5);
QUEUE,M+4;
SEIZE:CAR(M);
QUEUE,M+5;
SEIZE:SCAR(M);
BRANCH,2:
   ALWAYS,VECHIC:
   ALWAYS,MAKE;

VECHIC
ASSIGN:M=P(A(11),8);
QUEUE,M+7;
SEIZE:ROBOT(M);
ASSIGN:A(12)=P(A(11),9);
DELAY:ED(A(12));
RELEASE:ROBOT(M);
ASSIGN:M=P(A(11),5);
RELEASE:CAR(M);
ASSIGN:M=A(5);
RELEASE:PART(M),P(A(11),4):DISPOSE;
MAKE
\begin{align*}
\text{ASSIGN: } & A(1)=A(5); \\
\text{ASSIGN: } & A(2)=0; \\
\text{ASSIGN: } & A(4)=P(A(11),13); \\
\text{BACKMAK: } & A(2)=A(2)+1; \\
\text{BRANCH,1: } & \begin{cases} \\
\text{IF, } A(2) \cdot GT \cdot A(4), \text{OUT: } & \\
\text{ELSE, DOWNM1; } & \\
\end{cases} \\
\text{DOWNM1: } & \begin{cases} \\
\text{ALWAYS, BACKMAK: } & \\
\text{ALWAYS, DOWNM2; } & \\
\end{cases} \\
\text{DOWNM2: } & \begin{cases} \\
\text{ASSIGN: } & A(3)=2 \cdot A(2)+12; \\
\text{ASSIGN: } & A(5)=P(A(11),A(3)); \\
\text{ASSIGN: } & A(3)=A(3)+1; \\
\text{ASSIGN: } & A(6)=P(A(11),A(3)); \\
\text{ASSIGN: } & M=A(5); \\
\text{BRANCH,2: } & \begin{cases} \\
\text{ALWAYS, MAKEPT: } & \\
\text{ALWAYS, CHKPUL; } & \\
\end{cases} \\
\text{MAKEPT: } & \begin{cases} \\
\text{ASSIGN: } & A(12)=A(6) \cdot P(A(11),4); \\
\text{QUEUE, } M+12; \\
\text{SEIZE: PART(M), A(12); } \\
\text{QUEUE, } M+6; \\
\text{SEIZE: MCELL(M); } \\
\text{ASSIGN: } & A(12)=P(A(11),12); \\
\text{DELAY: ED((A(12)) \cdot P(A(11),4)); } \\
\text{ASSIGN: } & M=P(A(11),10); \\
\text{RELEASE: MCELL(M); } \\
\text{ASSIGN: } & A(12)=P(A(11),11); \\
\text{DELAY: ED((A(12))); } \\
\text{ASSIGN: } & M=P(A(11),5); \\
\text{RELEASE: SCAR(M): DISPOSE; } & \\
\end{cases} \\
\text{DOWNM3: } & \begin{cases} \\
\text{ASSIGN: } & M=P(A(11),10); \\
\end{cases} \\
\text{CHKPUL: } & \begin{cases} \\
\text{ASSIGN: } & A(5)=M; \\
\text{ASSIGN: } & A(11)=A(5)+3; \\
\text{BRANCH,1: } & \begin{cases} \\
\text{IF, } P(A(11),2) \cdot EQ \cdot 2, \text{PULL1: } & \\
\text{ELSE, OUT; } & \\
\end{cases} \\
\text{OUT: } & \text{DELAY: 0.0: DISPOSE; } \\
\end{cases}
\end{align*}
; *** Inspection station model ***
INSP
  BRANCH,1:
    WITH,A(4),CHECK:
    ELSE,PASS;
CHECK
  QUEUE,M;
  SEIZE:STATIONN(M);
  DELAY:ED(A(3));
  RELEASE:STATIONN(M);
BRANCH,1:
  WITH,A(5),REPAIR:
  ELSE,PASS;
REPAIR
  ASSIGN:M=A(B);
  QUEUE,M+8;
  SEIZE:REPAIR(M);
  DELAY:ED(A(7));
  RELEASE:REPAIR(M);
BRANCH,1:
  WITH,A(6),SCRAP:
  ELSE,PASS;
SCRAP
  DELAY:O:DISPOSE;
PASS
  DELAY:O:NEXT(LOOP);

; *** Task station model ***
TASK
  QUEUE,M;
  SEIZE:STATIONN(M);
  DELAY:ED(A(3));
  RELEASE:STATIONN(M):NEXT(LOOP);
Model-8-stations
Parameters of Example Model Model-8-Stations

Starting Point of Line
Name of line: Y
Interarrival time distribution: NORMAL
Mean: 100
Standard deviation: 5

Starting Point of Line
Name of line: X
Interarrival time distribution: EXPONENTIAL
Mean: 300

Starting Point of Line
Name of line: Z
Interarrival time distribution: NORMAL
Mean: 75
Standard deviation: 5

Final Product from Assembly Line
Part name: R
Capacity and initial inventory at the stock point:
Maximum number of parts at stock point: 2000
Initial number of parts at stock point: 8

Demand Stock Point
Part name: F
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK2
Moving time distribution: UNIFORM
Minimum: 6
Maximum: 14

Supply Stock Point
Part name: F
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK2
Moving time distribution: UNIFORM
Minimum: 6
Maximum: 14
Supply Stock Point
Part name: H
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK3
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Demand Stock Point
Part name: H
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK3
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Supply Stock Point
Part name: E
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Demand Stock Point
Part name: E
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Supply Stock Point
Part name: I
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK3
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Demand Stock Point
Part name: I
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK3
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Supply Stock Point
Part name: B
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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK2
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 14

Demand 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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1

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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1

Demand 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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1

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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1

Demand 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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1

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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1

Demand 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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1

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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Demand 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: 4
Initial number of carts at supply stock point: 4
Vehicle used to move carts between stock points: TRUCK1
Moving time distribution: UNIFORM
Minimum: 0
Maximum: 12

Push stock point
Part name: B
Capacity and initial inventory at the stock point:
Maximum number of parts at stock point: 2000
Initial number of parts at stock point: 120

Push stock point
Part name: B
Capacity and initial inventory at the stock point:
Maximum number of parts at stock point: 2000
Initial number of parts at stock point: 120

Ordered from outside
Part name: j
Capacity and initial inventory at the stock point:
Maximum number of parts at stock point: 10000
Initial number of parts at stock point: 10000
Will Part J be replenished during the simulation? No

Ordered from outside
Part name: k
Capacity and initial inventory at the stock point:
Maximum number of parts at stock point: 10000
Initial number of parts at stock point: 10000
Will Part K be replenished during the simulation? No

Ordered from outside
Part name: l
Capacity and initial inventory at the stock point:
Maximum number of parts at stock point: 10000
Initial number of parts at stock point: 10000
Will Part L be replenished during the simulation? No

Inspection Station
Station name: INSPI
Name of inspector: INSPECTOR
Name of repairman: REPAIRMAN
Name of plane for scrap parts: SCRAP
Inspection time distribution: NORMAL
Mean: 50
Standard deviation: 5
Repair time distribution: NORMAL
Mean: 400
Standard deviation: 10
Inspection rate (between 0 and 1): 1
Reject (repair) rate (between 0 and 1): 0.2
Scrap rate (between 0 and 1): 0.5

Task Station
Station name: TASK6
Task time distribution: NORMAL
Mean: 100
Standard deviation: 5

Task Station
Station name: TASK2
Task time distribution: NORMAL
Mean: 300
Standard deviation: 10

Assembly Station
Station name: ASSY5
Parts required for assembly:
Name of part #1: E
Number of part #1: 2
Assembly time distribution: NORMAL
Mean: 100
Standard deviation: 5

Assembly Station
Station name: ASSY1
Parts required for assembly:
Name of part #1: B
Number of part #1: 3
Name of part #2: C
Number of part #2: 2
Assembly time distribution: NORMAL
Mean: 300
Standard deviation: 10

Assembly Station
Station name: ASSY3
Parts required for assembly:
Name of part #1: D
Number of part #1: 4
Assembly time distribution: NORMAL
Mean: 300
Standard deviation: 10
Assembly Station
Station name: ASST8
Parts required for assembly:
Name of part #1: H
Number of part #1: 1
Assembly time distribution: NORMAL
Mean: 75
Standard deviation: 5

Assembly Station
Station name: ASST7
Parts required for assembly:
Name of part #1: F
Number of part #1: 2
Name of part #2: G
Number of part #2: 1
Assembly time distribution: NORMAL
Mean: 75
Standard deviation: 5

Manufacturing Cell — F
Cell name: MC2
Items required to make the part F:
Number of item types required: 1
Name of item #1: L
Number of item #1: 2
Setup time for a set of parts: CONSTANT
Constant: 0
Manufacturing time for a part: NORMAL
Mean: 10
Standard deviation: 1

Manufacturing Cell — E
Cell name: MC2
Items required to make the part E:
Number of item types required: 2
Name of item #1: J
Number of item #1: 2
Name of item #2: K
Number of item #2: 1
Setup time for a set of parts: CONSTANT
Constant: 0
Manufacturing time for a part: NORMAL
Mean: 10
Standard deviation: 1

Manufacturing Cell — I
Cell name: MC2
Items required to make the part I:
Number of item types required: 2
Name of item #1: J
Name of item #1: 2
Name of item #2: K
Name of item #2: 1
Setup time for a set of parts: CONSTANT
Constant: 0
Manufacturing time for a part: NORMAL
Mean: 5
Standard deviation: 1
Manufacturing Cell -- G
Cell name: NC2
Items required to make the part G:
Number of item types required: 1
Name of item #1: K
Number of item #1: 2
Setup time for a cart of parts: CONSTANT
Constant: 0
Manufacturing time for a part: NORMAL
Mean: 10
Standard deviation: 1

Manufacturing Cell -- H
Cell name: NC1
Items required to make the part H:
Number of item types required: 1
Name of item #1: I
Number of item #1: 1
Setup time for a cart of parts: CONSTANT
Constant: 0
Manufacturing time for a part: NORMAL
Mean: 30
Standard deviation: 3

Manufacturing Cell -- C
Cell name: NC1
Items required to make the part C:
Number of item types required: 1
Name of item #1: I
Number of item #1: 1
Setup time for a cart of parts: CONSTANT
Constant: 0
Manufacturing time for a part: NORMAL
Mean: 30
Standard deviation: 3
GPSS Code of Example Model Model-8-Stations

ANDY-TAYLOR:>dwan>amps>gpss-8-stations.lisp.2 3/18/89 02:04:38 Page 1

100 /* -------------------------------------------------------------------------
110     This is a 8-stations model.                     3/15/89
120 */
130 /* -------------------------------------------------------------------------
140 */
150 /* This is a GPSS program automatically created from
160 */
170 /* AMPS  -  (Automatic Manufacturing Programming System)
180 */
190 /* developed at
200 */
210 /* The University of Alabama in Huntsville, 1986.
220 */
230 */
240 /** ------------------------------------------------------------------------
250 */
260 SIMULATE
270 UNIFORM FUNCTION RHO, C2
280 /*---------------------------------------------------------------*/
290 PER1, FUNCTION RHO, C2
300 /* CHECKS */
310 PER1, FUNCTION RHO, C2
320 /* REPAIR */
330 PER1, FUNCTION RHO, C2
340 /* SCRAPP */
350 PER1, FUNCTION RHO, C2
360 /* TIME */
370 VARIABLE 6, 8FHS: UNIFORM
380 */
390 VARIABLE 6, 8FHS: HORN
400 */
410 VARIABLE 6, 8FHS: UNIFORM
420 */
430 VARIABLE 6, 8FHS: HORN
440 */
450 VARIABLE 6, 8FHS: UNIFORM
460 */
470 VARIABLE 6, 8FHS: HORN
480 */
490 VARIABLE 6, 8FHS: UNIFORM
500 */
510 VARIABLE 6, 8FHS: HORN
520 */
530 VARIABLE 6, 8FHS: UNIFORM
540 */
550 VARIABLE 6, 8FHS: HORN
560 */
570 VARIABLE 6, 8FHS: UNIFORM
580 */
590 VARIABLE 6, 8FHS: HORN
600 */
610 VARIABLE 6, 8FHS: UNIFORM
620 */
630 VARIABLE 6, 8FHS: HORN
640 */
650 VARIABLE 6, 8FHS: UNIFORM
660 */
670 VARIABLE 6, 8FHS: HORN
680 */
690 VARIABLE 6, 8FHS: UNIFORM
700 */
710 */
720 /* definition of matrix */
730 */
740 PART MATRIX .0, 12.2
750 */
760 */
770 */
780 */
790 */
800 */
810 */
820 */
830 */
840 */
850 */
860 */
870 */
880 */
890 */
900 */
910 */
920 */
930 */
940 */
950 */
960 */
970 */
980 */
990 */
1000 */
1010 */
1020 */
1030 */
1040 */
1050 */
1060 */
1070 */
1080 */
1090 */
1100 */
1110 */
1120 */
1130 */
1140 */
1150 */
1160 */
1170 */
1180 */
1190 */
1200 */
1210 */
1220 */
1230 */
1240 */
1250 */
1260 */
1270 */
1280 */
1290 */
1300 */
1310 */
1320 */
1330 */
1340 */
1350 */
1360 */
1370 */
1380 */
1390 */
1400 */
1410 */
1420 */
1430 */
1440 */
1450 */
1460 /* capacity of part & cart counters */
1470 PA.R STORAGE 2000
1480 PA.F STORAGE 4
1490 CART.F STORAGE 8
1500 SCART.F STORAGE 8
1510 PA.M STORAGE 4
1520 CART.M STORAGE 8
1530 SCART.M STORAGE 8
1540 PA.E STORAGE 4
1550 CART.E STORAGE 8
1560 SCART.E STORAGE 8

D-10

ORIGINAL PAGE IS
OF POOR QUALITY
ANDY-TAYLOR::dwang::amps::gps::stations::lisp::p2

D-12
Andy Taylor

2170 MAKEVALUE TIME,1,1,$TIME9 ; Inspection time of INSPI4 is TIME9
2180 MAKEVALUE TIME,1,2,$TIME10 ; Inspection time of INSPI4 is TIME10
2190 *** PART (12) REQUIRED AT EACH STATION ***
2200 MAKEVALUE STAB,5,2,1 ; Station ASSY3 requires 1 part type(s).
2210 MAKEVALUE STAB,5,3,3 ; part E.
2220 MAKEVALUE STAB,5,4,1 ; 2 unit(s).
2230 MAKEVALUE STAB,1,2,2 ; 2 unit(s).
2240 MAKEVALUE STAB,1,3,1 ; 1 unit(s).
2250 MAKEVALUE STAB,1,4,3 ; 3 unit(s).
2260 MAKEVALUE STAB,1,5,6 ; part C.
2270 MAKEVALUE STAB,1,6,2 ; 2 unit(s).
2280 MAKEVALUE STAB,1,7,1 ; part D.
2290 MAKEVALUE STAB,1,9,12 ; 1 unit(s).
2300 MAKEVALUE STAB,1,9,4,4 ; 4 unit(s).
2310 MAKEVALUE STAB,1,9,5,1 ; Station ASSY9 requires 1 part type(s).
2320 MAKEVALUE STAB,2,1,2 ; 2 unit(s).
2330 MAKEVALUE STAB,2,4,1 ; 1 unit(s).
2340 MAKEVALUE STAB,2,7,2 ; 2 unit(s).
2350 MAKEVALUE STAB,2,8,2 ; 2 unit(s).
2360 MAKEVALUE STAB,2,9,7 ; part E.
2370 MAKEVALUE STAB,2,9,7 ; part F.
2380 *** SUPPLY SYSTEM OF EACH PART ***
2390 MAKEVALUE PART,1,2,1 ; part A is in push mode
2400 MAKEVALUE PART,1,2,2 ; part B is in pull mode
2410 MAKEVALUE PART,1,2,3 ; part C is in pull mode
2420 MAKEVALUE PART,1,2,4 ; part D is in pull mode
2430 MAKEVALUE PART,1,2,5 ; part E is in pull mode
2440 MAKEVALUE PART,1,2,6 ; part F is in pull mode
2450 MAKEVALUE PART,1,2,7 ; part G is in push mode
2460 MAKEVALUE PART,1,2,8 ; part H is in push mode
2470 MAKEVALUE PART,1,2,9 ; part I is ordered from outside
2480 MAKEVALUE PART,1,2,10 ; part J is ordered from outside
2510 *** CART COUNTER AT EACH DESTINATION ***
2520 MAKEVALUE SCART,1,1,SCART_F
2530 MAKEVALUE SCART,1,1,SCART_H
2540 MAKEVALUE SCART,1,1,SCART_E
2550 MAKEVALUE SCART,1,1,SCART_1
2560 MAKEVALUE SCART,1,1,SCART_0
2570 MAKEVALUE SCART,1,1,SCART_G
2580 *** CART COUNTER AT SOURCE ***
2590 MAKEVALUE SCART,1,1,SCART_F
2600 MAKEVALUE SCART,1,1,SCART_H
2610 MAKEVALUE SCART,1,1,SCART_E
2620 MAKEVALUE SCART,1,1,SCART_I
2630 MAKEVALUE SCART,1,1,SCART_G
2640 MAKEVALUE SCART,1,1,SCART_C
2650 *** UNLOADING TO MOVE PARTS ***
2660 MAKEVALUE FIG,2,1,FIG2 ; part F is transported by TRUCK2
2670 MAKEVALUE FIG,5,1,FIG3 ; part H is transported by TRUCK3
2680 MAKEVALUE FIG,5,1,FIG3 ; part E is transported by TRUCK3
2700 MAKEVALUE FIG,6,1,FIG1 ; part I is transported by TRUCK1
2710 MAKEVALUE FIG,7,1,FIG2 ; part G is transported by TRUCK2
2720 MAKEVALUE FIG,6,1,FIG1 ; part C is transported by TRUCK1
2730 TERMINATE
2740 ****************************************
2750 //= ASSEMBLY LINE Y
2755 ****************************************
2760 GENERATE VTIME11
2770 ASSIGN 2,5 ; Station 3 is ASSYS
2780 TRANSFER SBR,ASM,TRM1
2790 ASSIGN 2,6 ; Station 6 is TASK6
2800 TRANSFER SBR,ASK,TRM1
2810 ENTER PR,0,1
2820 TERMINATE
2830 ****************************************
2840 //= ASSEMBLY LINE X
2840 ****************************************
2850 GENERATE VTIME1
2850 ASSIGN 2,1 ; Station 1 is ASSY1
2860 TRANSFER SBR,ASM,TRM1
2870 TERMINATE
2880 ****************************************
2890 D-13

Original page is of poor quality.
ANDY-TAYLOR: >> dwan > amps > gpss-8-stations.lisp 2

2900 ASSIGN 2, 2 ; station 2 is TASK2
2910 TRANSFER GR, TASK2, TRL1
2920 ASSIGN 2, 3 ; station 3 is ASSY3
2930 TRANSFER GR, ASSY3, TRL1
2940 ASSIGN 2, 4 ; station 4 is INSP4
2950 TRANSFER GR, INSP4, TRL1
2960 ENTER PM, 1
2970 TERMINATE

2980 # ASSEMBLY LINE 2

3000 # ASSEMBLY LINE 2

3010 GENERATE PM1, NE12
3020 ASSIGN 2, 7 ; station 7 is ASSY7
3030 TRANSFER GR, ASSY7, TRL1
3040 ASSIGN 2, 0 ; station 8 is ASSY8
3050 TRANSFER GR, ASSY8, TRL1
3060 ENTER PM, 1
3070 TERMINATE

3090 # INSPECTION STATION

3100 # INSPECTION STATION

3110 INSP ASSIGN 3, NHISTAR(P2, 2) ; inspection station
3120 ASSIGN 4, NHIPERC(P3, 1) ; inspection rate
3130 ASSIGN 5, NHISTIME(P3, 1) ; time for inspection
3140 ASSIGN 6, NHISTIME(P3, 2) ; time for repair
3150 QUEUE NHISTAR(P2, 1) ; count entering parts
3160 DEPART NHISTAR(P2, 1) ; and pass
3170 TRANSFER FMH4 ; to be checked
3180 CHECK NHISTAR(P3, 1) ; wait on the insp. facility
3190 GEIZE NHISTAR(P3, 1) ; seize the insp. facility
3200 DEPART NHISTAR(P3, 1) ; leave the insp. queue
3210 ADVANCE V65 ; inspecting part
3220 RELEASE NHISTAR(P3, 1) ; release the insp. facility
3230 ASSIGN 4, NHIPERC(P3, 2) ; repair rate
3240 TRANSFER FMH4 ; to be repaired?
3250 REPAIR QUEUE NHISTAR(P3, 2) ; wait on repairing facility
3260 GEIZE NHISTAR(P3, 2) ; seize the repairing facility
3270 DEPART NHISTAR(P3, 2) ; leave the queue
3280 ADVANCE V66 ; repairing parts
3290 RELEASE NHISTAR(P3, 2) ; release the repairing facility
3300 ASSIGN 4, NHIPERC(P3, 3) ; scrap rate
3310 TRANSFER FMH4 ; if part scrapped?
3320 PASS TRANSFER P, TRL1, 1 ; finish inspection and return
3330 SCRAP QUEUE NHISTAR(P3, 3) ; count scrapped parts
3340 DEPART NHISTAR(P3, 3) ; and pass
3350 TERMINATE ; terminate the transaction

3360 # TASK STATION

3370 # TASK STATION

3380 TASK ASSIGN 3, NHISTAR(P2, 1) ; name of the task station
3390 ASSIGN 6, NHISTIME(P2, 1) ; time for operation
3400 QUEUE P3 ; wait on the facility
3410 GEIZE P3 ; seize the facility
3420 DEPART P3 ; leave the queue
3430 ADVANCE V46 ; perform operation
3440 RELEASE P3 ; release the facility
3450 TERMINATE ; finish and return

3460 # ASSEMBLY STATION

3470 # ASSEMBLY STATION

3480 ASH ASSIGN 3, NHISTAR(P2, 1) ; name of the station
3490 ASSIGN 7, NHISTAR(P2, 2) ; no. of part types required
3500 ASSIGN 6, NHISTIME(P2, 1) ; assembly time
3510 ASSIGN 0, 1 ; index for part type
3520 ASSIGN 9, 2 ; index for units of part
3530 QUEUE P3 ; wait on the facility
3540 PRO ASSIGN 0, 2 ; point to next type
3550 ASSIGN 9, 3 ; point to next no. of units
3560 ASSIGN 5, NHISTAR(P2, P3) ; part id
3570 ASSIGN 18, NHIPART(P2, 1) ; name of part
3580 ASSIGN 20, NHISTAR(P2, P3) ; number of units of part
3590 QUEUE P10 ; wait to get parts
3600 TRANSFER GR, TAKEP, TRL2 ; get parts

3610 #
ANDY-TAYLOR::>dwan>amps gpss-8-stations.lisp.2 3/18/89 02:04:38 Page 6

3630 DEPART P10 leave the queue
3640 LOOP P,PA0 received all parts req'd?
3650 DECREASE P3 decrease the facility
3660 DEPART P3 leave the queue
3670 ADVANCE U46 assembling parts
3680 RELEASE P3 release the facility
3690 TRANSFER P,SRN1,1 return to the calling statement

3700 # INVENTORY CONTROL
3710
3720 TAKEP TEST E HANDPRT(PS,2),1,PULL pull or push node?
3730 PUSH TEST GE 810,P20 available => req'd
3740 LEAVE 810,P20 take units requested
3750 TRANSFER P,SRN2,1 return
3760 PULL ASSIGN 38,NHSIZE(PS,1) none of cart for the part
3770 TEST GE 810,P20,NEEDC available => requested?
3780 LEAVE 810,P20 take units requested
3790 SPLIT 1,USEP signal use of parts
3800 TRANSFER P,SRN2,1 return
3810 NEEDC ASSIGN 20,810 requested - available => unfilled
3820 LEAVE 810,P20 take remaining
3830 SPLIT 1,USEP signal use of parts
3840 TEST GE 810,P20,1 any full cart available?
3850 LEAVE 810 take 1 cart
3860 ENTER 810,NHSIZE(PS,1) make parts available
3870 TEST GE 810,P20,NEEDC available => unfilled?
3880 LEAVE 810,P20 take units unfilled
3890 SPLIT 1,USEP signal use of parts
3900 TRANSFER P,SRN2,1 return
3910 USEP TEST GE 810,0,EMPTY check if cart empty
3920 TERMINATE terminate the transaction
3930 EMPTY SPLIT 1,ORDER1 signal to order
3940 TEST GE 810,1 any full cart available
3950 LEAVE 810 take 1 cart
3960 ENTER 810,NHSIZE(PS,1) make parts available
3970 TERMINATE terminate the transaction
3980 ORDER1 ASSIGN 26,NHSIZE(PS,1) none of the vehicle
3990 ASSIGN 36,NHSIZE(PS,1) moving time
4000 QUEUE P26 wait on the vehicle
4010 DECREASE P26 seize the vehicle
4020 DEPART P26 leave the queue
4030 ADVANCE U36 moving the empty cart
4040 RELEASE P26 release the vehicle
4050 SPLIT 1,GETIF signal to get a full cart
4060 ASSIGNS 12,PS assign part id
4070 ASSIGN 15,NHSIZE(PS,1) assign none of the cart
4080 GETIF TRANSFER SBM,RSR,RMHS order a cart of parts
4090 ENTER P15,1 receive a cart
4100 TERMINATE terminate the transaction
4110 GETIF ASSIGN 31,NHSIZE(PS,1) index for a full cart
4120 QUEUE P31 wait on a full cart
4130 TEST GE 831,1 whether a full cart is available
4140 LEAVE 831 take 1 cart
4150 DEPART P31 leave the queue
4160 BENDIF QUEUE P26 wait on the vehicle
4170 DECREASE P26 seize the vehicle
4180 DEPART P26 leave the queue
4190 ADVANCE U36 moving the full cart
4200 RELEASE P26 release the vehicle
4210 TERMINATE increase inventory by 1 cart
4220 TERMINATE terminate the transaction

4230 # MANUFACTURING CELL
4240
4250
4260
4270 NFC ASSIGN 13,NHSIZE(P12,1) none of nucart's cart
4280 ASSIGN 14,NHSIZE(P12,1) setup time of a cart
4290 ASSIGN 16,NHSIZE(P12,2) move to the facility
4300 QUEUE P18 wait on the facility
4310 ASSIGN 7,NHSIZE(P12,1) move of parts to a cart
4320 CARTO ASSIGN 17,NHSIZE(P12,1) none of item types req'd
4330 ASSIGNS 8,0 index for item type
4340 ASSIGN 9,1 index for units of item
4350 PARTO ASSIGN 8,2 points to next type

ORIGINAL PAGE IS OF POOR QUALITY
<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4360</td>
<td>ASSIGN</td>
<td>9.2</td>
<td>Point to next no. of units</td>
</tr>
<tr>
<td>4370</td>
<td>ASSIGN</td>
<td>5,REITEM(P12,P9)</td>
<td>iid of the item req'd</td>
</tr>
<tr>
<td>4380</td>
<td>ASSIGN</td>
<td>18,REITEM(P5,1)</td>
<td>Name of the item req'd</td>
</tr>
<tr>
<td>4390</td>
<td>ASSIGN</td>
<td>29,REITEM(P12,P9)</td>
<td>Units of the item req'd</td>
</tr>
<tr>
<td>4400</td>
<td>QUEUE</td>
<td>P18</td>
<td>Wait on the item</td>
</tr>
<tr>
<td>4410</td>
<td>TRANSFER</td>
<td>GBR,TAKEP,TINH2</td>
<td>Get items</td>
</tr>
<tr>
<td>4420</td>
<td>DEPART</td>
<td>P18</td>
<td>Leave the queue</td>
</tr>
<tr>
<td>4430</td>
<td>LOOP</td>
<td>17,FRG0</td>
<td>Loop for next item type req'd</td>
</tr>
<tr>
<td>4440</td>
<td>LOOP</td>
<td>7,FRG0</td>
<td>Loop for next part to be node</td>
</tr>
<tr>
<td>4450</td>
<td>FAC</td>
<td>SEIZE P13</td>
<td>Seize the facility</td>
</tr>
<tr>
<td>4460</td>
<td>DEPART</td>
<td>P13</td>
<td>Leave the queue</td>
</tr>
<tr>
<td>4470</td>
<td>ADVANCE</td>
<td>V+14</td>
<td>Register facility</td>
</tr>
<tr>
<td>4490</td>
<td>MTIME</td>
<td>FVARIABLE</td>
<td>Manufacturing time</td>
</tr>
<tr>
<td>4500</td>
<td>RELEASE</td>
<td>P13</td>
<td>Release the facility</td>
</tr>
<tr>
<td>4510</td>
<td>TRANSFER</td>
<td>P,FRH3,1</td>
<td>Manufacturing complete</td>
</tr>
</tbody>
</table>

D-16