Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

SLAM II®: A TUTORIAL

Jean J. O'Reilly
Pritsker & Associates, Inc.
P.O. Box 2413
West Lafayette, IN 47906

INTRODUCTION

SLAM II, the Simulation Language for Alternative Modeling, was the first simulation language which allowed a modeler to formulate a system description using any of three approaches (world views) or any combination of the three. SLAM II integrates the process-oriented world view of Q-GERT®[2] and the discrete-event/continuous world views of GASP IV™[3] in order to free the modeler to select the approach which best represents his system or with which he feels most comfortable. This integrated framework allows the SLAM II user to take advantage of the simplicity of the process-oriented (network) approach and to extend a model with discrete event constructs should the network approach become too restrictive. Continuous variables may be used in conjunction with a network or discrete event model whenever this is the most convenient way to represent system elements. The ability to construct combined network-event-continuous models with interactions between each orientation makes SLAM II an extremely flexible tool for simulation. In order to illustrate the process of modeling with SLAM II, this paper presents a simple model of a manufacturing process. The model is developed first with the network approach, then embellished with discrete and continuous elements.

NETWORK MODELING

A simulation model normally begins with a network, or flow diagram, which graphically portrays the flow of entities (people, parts, or information, for example) through the system. A SLAM II network is made up of "nodes" at which processing is performed. Twenty node types in SLAM II, shown in Figure 1, provide for such functions as entering or exiting the system, seizing or freeing a resource, changing variable values, collecting statistics, and starting or stopping entity flow based on system conditions. Nodes are connected by branches, called "activities", which define the routing of the entities through the system. Routing may be deterministic, probabilistic, or based on system variables. Time delays on activities may represent processing times, travel times, or waiting times. Entities which proceed from node to node over activities may have unique characteristics, called "attributes", which control their processing. Entities may reside in "files", or ordered lists of entities which are waiting for some change in system status. The graphical framework for representing a network model simplifies model development and communication.

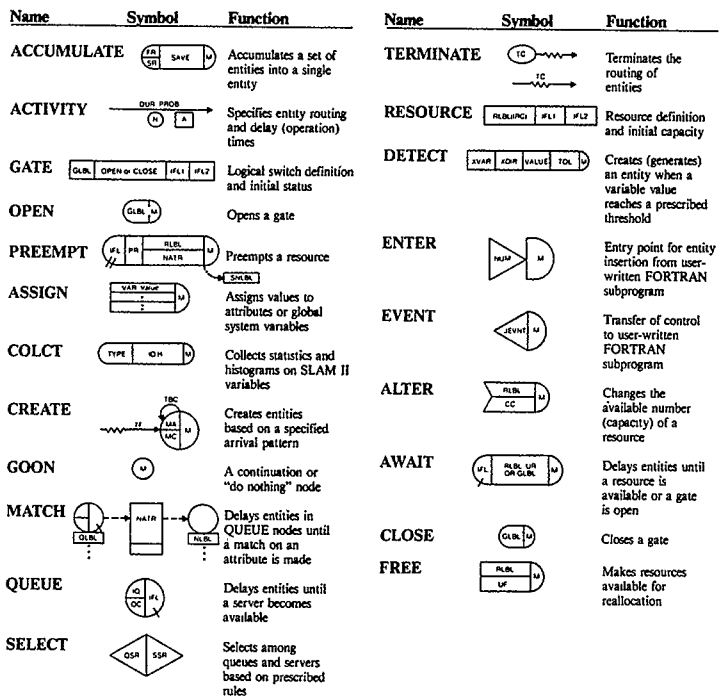


Figure 1: SLAM II Network Symbols

The example model which follows illustrates some of the network processes, system variables, and random variables which are available in SLAM II. This model simulates items in the final stage of production, where they move through a finishing station and a testing station before being routed to packing. At the testing station there is room for at most 4 units, including the unit being tested. Any backlog requiring more than 4 units at the testing station will block the preceding finishing station. Ninety percent of the tested items will pass the test procedure; ten percent will fail and require calibration. Calibration is performed at the testing station.

The testing station is subject to breakdowns, during which time it can no longer process items. When a breakdown occurs, an item being processed at the testing station is held until repairs are complete. It then resumes processing from the point of interruption.

A SLAM II model of this system is in two segments, shown in Figure 2. The first segment models the movement of items through the two stations. It begins with a CREATE node which generates the first arrival of an item at simulated time 0. and continues to generate arrivals at a rate drawn from a normal distribution.

Upon arrival to the system, the item enters a QUEUE node where it waits for the finishing station in file 1. The finishing operation is represented by an ACTIVITY, or branch, following the QUEUE; finishing time is sampled from a triangular distribution. Activities which emanate from QUEUE nodes are known as "service" activities, and the number of entities which may be served concurrently is limited by the number of servers specified. In this example, the finishing station is modeled as a single server.

The testing station is modeled as a resource (named TESTER) rather than a server, since it may be required for more than one activity and for convenience in modeling breakdowns. After finishing, an item enters an AWAIT node to be allocated the TESTER resource. At most 3 items can be waiting for the TESTER; when the buffer is full the finishing station will be blocked (shown by two parallel lines). Once the resource is available, an item enters the testing activity, where the duration is a sample from a normal distribution with a mean of .5 hours.

The testing activity ends at a GOON node which serves as a branching point. The tested item will take one of the activities emanating from the GOON node. Ten percent of the tested items will require calibration, which takes an average of .75 hours, normally distributed. Ninety percent will proceed without delay. Both branches end at a FREE node which makes the TESTER available to other waiting entities. The item which has completed processing then passes through a COLCT node which records the time interval between current simulated time and the item's time of entry, saved in its first attribute. The item then exits the system.

The second network segment models the breakdowns of the testing station. It begins with a CREATE node which generates one breakdown entity. This entity is routed to a PREEMPT node after a delay time representing the time between breakdowns. At the PREEMPT node, the entity captures the resource TESTER. If TESTER is performing a finishing or calibration operation at the time of the breakdown, the item being processed will be held until the resource becomes available once again. Processing will then resume at the point of interruption.

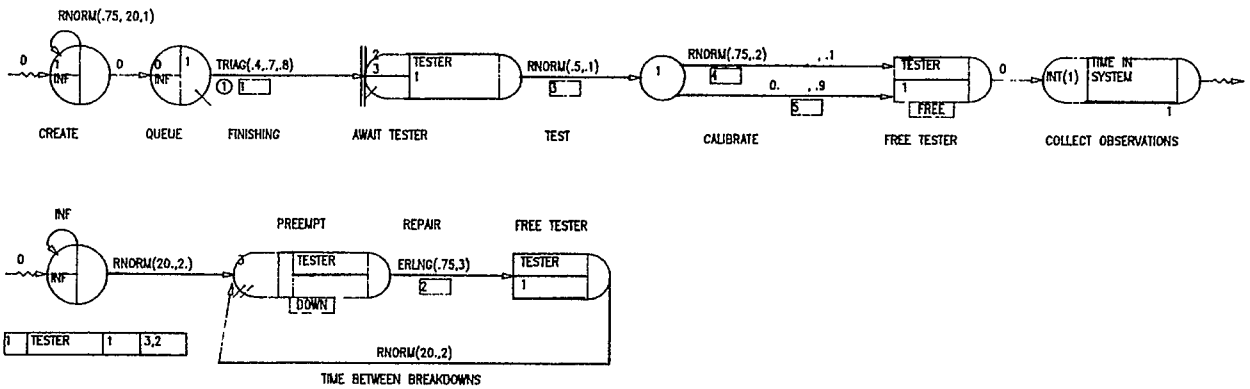


Figure 2: SLAM II Network Model

From the PREEMPT node, the breakdown entity enters the repair ACTIVITY. Repair time is sampled from an Erlang distribution. Following repair, the tester resource is made available by routing the entity through a FREE node. The next breakdown is then scheduled by routing the entity back to the PREEMPT node with an activity duration representing the time between breakdowns.

The input corresponding to this network model is shown in Figure 3. Each symbol corresponds to an input statement, and each statement may be followed by a comment which describes the processing being performed. The output from this model would automatically report utilization statistics for the finishing and testing stations, queuing statistics for both stations, time in system statistics, and throughput counts.

USING DISCRETE EVENT CONCEPTS

In the discrete event orientation of SLAM II, the modeler identifies the discrete points in time at which the state of the system can change and develops the logic associated with each such "event". SLAM II provides support subroutines which perform such common simulation tasks as scheduling events, moving entities into and out of files, collecting statistics, and obtaining random samples. Most models built with SLAM II are not strictly network or discrete event but a combination of the two approaches.

To illustrate the inclusion of discrete event constructs in a network model, let us modify the network model discussed above. Assume that when a breakdown occurs the percentage of breakdowns per unit time is investigated. If this percentage is greater than 4.5%, the testing station will be taken out of service for maintenance, adding eight hours to the repair time. Should this occur, any items waiting to be finished will be subcontracted outside the system. In a SLAM II network, there is no way to exit a QUEUE node until the server on the emanating activity is available. The subcontracting will therefore be modeled with a user-written event at each breakdown occurrence.

The addition of this logic, shown in Figure 4, involves the addition of an EVENT node following the PREEMPT node which models a breakdown. Upon the arrival of an entity to an EVENT node, control is passed to the user-written subroutine EVENT. The code for subroutine EVENT for this model is shown in Figure 5. Within this subroutine, the number of breakdowns per unit time is calculated using the SLAM II function NNCNT (the number which have completed an activity) and the SLAM II variable TNOW (current simulated time). The SLAM II variable ATRIB(1) is used to reset the first attribute of the breakdown entity, which is initially zero. If BPUT (breakdowns per unit time) is within the acceptable range, no further processing is done. Otherwise ATRIB(1) is set to 8, the time required for maintenance. The number of items waiting for finishing is obtained from the SLAM II function NNQ (number in queue). Any items currently in the queue are removed using the SLAM II subroutine ULINK. Finally, the occurrence of a maintenance delay and the number of subcontracted units is recorded by a call to the SLAM II subroutine COLCT.

CONTINUOUS MODELING

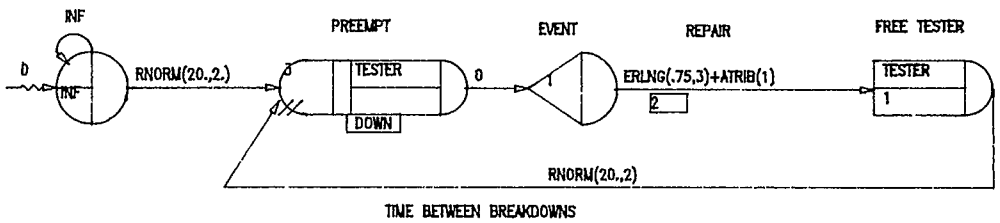
In a continuous simulation model the state of the system is represented by variables that change continuously over time. The modeler specifies equations which determine the values of state variables and the "step size", or time increment, between the updating of variable values. These equations may be differential equations, in which case the simulator uses a numerical integration algorithm to obtain new variable values from the derivative values.

For continuous simulations, SLAM II provides a set of special storage arrays which the modeler uses to encode the equations of continuous variables. In combined models these variables (known as SS and DD variables) may be affected by the occurrence of discrete events as well as by their defining equations. To illustrate this interaction, let us suppose that in our sample model the breakdowns do not occur at discrete points in time, but are a function of the wear on the testing station. The

```
NETWORK;
      RESOURCE/TESTER(1),3,2;                    DEFINE RESOURCE
      CREATE,RNORM(.75,.20,1),,1;                CREATE ARRIVALS
      QUE(1);                                    FINISHING QUEUE
      ACT/1,TRIAG(.4,.7,.8);                     FINISHING
      AWAIT(2/3),TESTER/1,BLOCK;                 SEIZE THE TESTER
      ACT/3,RNORM(.5,.1);                        TESTING
      GOON,1;
      ACT/4,RNORM(.75,.2),.1,FREE;               10% NEED CALIBRATION
      ACT/5,,.9;                                 90% PASS TESTING
FREE  FREE,TESTER;                               FREE THE TESTER
      COLCT,INT(1),TIME IN SYSTEM,20/0./.25;     COLLECT STATISTICS
      TERM;                                      EXIT

      CREATE;                                    CREATE 1ST BREAKDOWN
      ACT,RNORM(20.,2.);                         DELAY BY TIME BETWEEN
DOWN  PREEMPT(3),TESTER;                         PREEMPT THE TESTER
      ACT/2,ERLNG(.75,3.);                       DOWN TIME
      FREE,TESTER;                               RELEASE THE TESTER
      ACT,RNORM(20.,2.),,DOWN;                   TIME BETWEEN FAILURES
      END;
```

Figure 3: SLAM II Input




```
;
        CREATE;
        ACT,RNORM(20.,2.);
DOWN    PREEMPT(3),TESTER;
        EVENT,1
        ACT/2,ERLNG(.75,3.)+ATRIB(1);
        FREE,TESTER;
        ACT,RNORM(20.,2.),,DOWN;
```

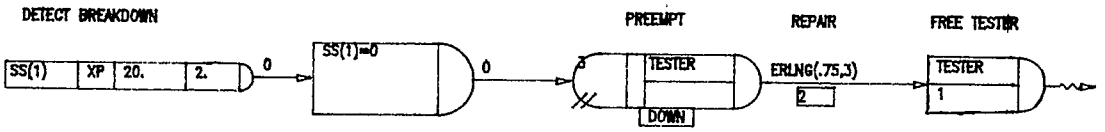
Figure 4: Augmented Model Network

```
      SUBROUTINE EVENT(I)
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
C  INITIALIZE MAINTENANCE TIME TO ZERO
C
      ATRIB(1) = 0.
C
C  CALCULATE BREAKDOWNS PER UNIT TIME
C
      XN = NNCNT(2)+1
      BPUT = XN/TNOW
      IF (BPUT .LT. .045) RETURN
C
C  BREAKDOWN PERCENTAGE GREATER THAN 4.5%.  SET MAINTENANCE TIME
C  TO EIGHT HOURS, SUBCONTRACT UNITS WAITING FOR FINISHING, AND
C  COLLECT STATISTICS ON THE NUMBER OF SUBCONTRACTS.
C
      ATRIB(1) = 8.
      NQ = NNQ(1)
      IF (NQ .GT. 0) THEN
        DO 10 I=1,NQ
        CALL ULINK(1,1)
   10 CONTINUE
      ENDIF
      XNQ = NQ
      CALL COLCT(XNQ,1)
      RETURN
      END
```

Figure 5: Augmented Model Event Routine

rate of wear on the tester is equal to the status of the testing/calibration operations (zero if idle, one if operating). Also suppose that the tester is subject to breakdown after 20 to 22 hours of use.

To implement this change, the breakdown subnetwork is replaced with the network segment shown in Figure 6. The FORTRAN associated with this representation of the system is subroutine STATE, also shown in Figure 6, in which DD(1), the derivative with respect to time of SS(1), is defined as the status of activity 2 (testing) plus the status of activity 3 (calibration). SLAM II will automatically integrate the value of SS(1) over time. The new network segment begins with a DETECT node which specifies that the node is to be activated when the value of SS(1) increases above 20. with a tolerance of 2. At that time a breakdown entity will be generated which will preempt the TESTER resource, reset tool wear to zero, and hold the resource until repairs are complete.




```
;
      DETECT,SS(1),XP,20.,2;                    DETECT WEAR THRESHOLD
      ASSIGN,SS(1)=0;                           RESET WEAR TO ZERO
DOWN  PREEMPT(3),TESTER;                        PREEMPT THE TESTER
      ACT/2,ERLNG(.75,3.);                      DOWN TIME
      FREE,TESTER;                              RELEASE THE TESTER
      TERM;

   SUBROUTINE STATE
   COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
  1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
   DD(1) = NNACT(3)+NNACT(4)
   RETURN
   END
```

Figure 6: Modifications for Continuous Modeling

APPLICATIONS

SLAM II has been distributed since 1979 by Pritsker & Associates, Inc. It is written in ANSI Standard FORTRAN, allowing it to be installed on any supermini or mainframe computer having a standard FORTRAN compiler. Annual revisions throughout the life of the program have significantly increased its flexibility and efficiency. SLAM II's FORTRAN base makes it easy to have a model include interfaces to other functions such as database management, statistical analysis, or linear optimization.

In 1984 a microcomputer version of SLAM II was made available from Pritsker & Associates for the IBM PC and compatible micros. The PC version produces, in addition to standard output reports, DIF files for use with other microcomputer software such as graphics and spreadsheets. Also in 1984, Pritsker & Associates announced the release of TESS™(The Extended Simulation System), a support system for simulation projects using SLAM II. Using a TESS database, manipulated with an easy-to-use command language, one may interactively build SLAM II models and prepare output reports from simulation data in a variety of graphical formats, including animation.

SLAM II has been used for hundreds of simulation projects and as the basis for simulation courses in many colleges and universities. Published applications (see references) describe models dealing with problems in manufacturing, transportation, material handling, staffing, experimental design, and many more. In other applications, SLAM II has been used to simulate computer and communications systems.

REFERENCES

1. Pritsker, A.A.B., Introduction to Simulation and SLAM II, Systems Publishing Corp., 1984.

2. Pritsker, A.A.B., Modeling and Analysis Using Q-GERT Networks, Halsted Press and Pritsker & Associates, 1977.

3. Pritsker, A.A.B., The GASP IV Simulation Language, John Wiley, 1974.

4. Pritsker, A.A.B. and C.E. Sigal, Management Decision Making: A Network Simulation Approach, Prentice-Hall, 1983.

5. "Simulation Modeling as an Aid to Casting Plant Design for a Aluminum Smelter", J.R. Gross, S. M. Hare, & Sukumar Roy, Presented at the IMACS Conference, Montreal, Canada, August, 1982.

6. "On the Usefulness of SLAM II for the Modeling and Simulation of Large Transport Systems", Andre Graber & Francois E. Cellier, Presented at the IMACS Conference, Montreal, Canada, August, 1982.

7. "A Simulation-Optimization Model for Exploration and Exploitation of Exhaustible Mineral Resources", David L. Martin, Presented at the SME-AIME Annual Meeting, Atlanta, GA, March, 1983.

8. "Analysis of Space Shuttle Ground Operations", J.R. Wilson, D.K. Vaughan, E. Naylor & R.G. Voss, Simulation, June 1982, pp.187-203.

9. "Application of Simulation and Zero-One Programming for Analysis of Numerically Controlled Machining Operations in the Aerospace Industry", W.R. Lilegdon, C.H. Kimpel & D.H. Turner, Presented at the Winter Simulation Conference, San Diego, CA, December, 1982.

10. "Applications of SLAM", A.A.B. Pritsker, IIE Transactions, March 1982, pp.70-77.

11. "SLAM II Model of the Rose Bowl Staffing Plans", D. Erdbruegger, D. Starks & C. Farris, Presented at the Winter Simulation Conference, San Diego, CA, December, 1982.

12. "PATRIOT Air Defense Weapon System Deployment Model Using SLAM", W.T. Prestwood, U.S. Army Missile Command Logistics Support Analysis Office, Redstone Arsenal, AL.

13. "An Integrated Model of Drilling Vessel Operations", S.E. Hoffman, M.M. Crawford & J.R. Wilson, Presented at the Winter Simulation Conference, Arlington, VA, December, 1983.

14. "Using SLAM and SDL®to Assess Space Shuttle Experiments", C.R. Standridge & J.R. Phillips, Simulation, July 1983, pp. 25-35.

15. "Simulation/Optimization of a Specialties Plant", R.M. Felder, P.M. Kester & J.M. McConney, Chemical Engineering Progress, June 1983, pp. 84-89.

16. "Operations Simulation for the Design of a Future Space Transportation System", W.D. Morris, T.A. Talay & D.G. Eide, Presented at the AIAA 21st Aerospace Sciences Meeting, Reno, Nevada, January 1983.

17. "A Critical Evaluation of Some Problems Associated with Clinical Caries Trials by Computer Simulation", K.H. Lu & L. Van Winkle, Journal of Dental Research, May 1984, pp. 796-804.

18. "A Simulation Study with a Combined Network and Production Systems Model of Pilot Behavior on an ILS-Approach", B. Doring & A. Knauper, Automatica, Vol. 19, 1983.

19. "Optimal Prepositioning of Empty Freight Cars", L.L. Ratcliffe, B. Vinod & F.T. Sparrow, Simulation, June 1984.

20. "Applications of Simulation: Combined Models", S.D. Duket & C.R. Standridge, Modeling, Issue No. 19, December, 1983.

21. "Simulation Models for Logistics Managers", J.N. McCallum & B.B. Nickey, Logistics Spectrum, Volume 18, Number 4, Winter 1984.

JEAN J. O'REILLY is a Senior Systems Consultant at Pritsker & Associates, Inc. She holds a Bachelor of Arts degree in Mathematics from St. Mary's College, Notre Dame, Indiana and a Master of Science in Applied Mathematics from Purdue University. Since joining P&A in 1978, Ms. O'Reilly has been involved in software development and in applying SLAM II in various consulting projects. In her current position as SLAM II product leader she is responsible for product development, user support, and training for the language.

Pritsker & Associates, Inc.
1305 Cumberland Avenue
P.O. Box 2413
West Lafayette, IN 47906
(317) 463-5557