# SLAM TUTORIAL

A.Alan B.Pritsker
Claude Dennis Pegden

## INTRODUCTION

SLAM is a simulation language that allows for alternative modeling approaches. It allows systems to be viewed from a process, event, or state variable perspective. These alternate modeling world views are combined in SLAM to provide a unified systems modeling framework (1).

In SLAM, a discrete change system can be modeled within an event orientation, process orientation, or both. Continuous change systems can be modeled using either differential or difference equations. Combined discrete-continuous change systems can be modeled by combining the event and/or process orientation with the continuous orientation. In addition, SLAM incorporates a number of features which correspond to the activity scanning orientation.

The process orientation of SLAM employs a network structure comprised of specialized symbols called nodes and branches in a manner similar to Q-GERT (2). These symbols model elements in a process such as queues, servers, and decision points. The modeling task consists of combining these symbols into a network model which pictorially represents the system of interest. In short, a network is a pictorial representation of a process. The entities in the system (such as people and items) flow through the network model. The pictorial representation of the system is transcribed by the modeler into an equivalent statement model for input to the SLAM processor.

In the event orientation of SLAM, the modeler defines the events and the potential changes to the system when an event occurs. The mathematical-logical relationships prescribing the changes associated with each event type are coded by the modeler as FORTRAN subroutines. A set of standard subprograms is provided by SLAM for use by the modeler to perform common discrete event functions such as event scheduling, file manipulations, statistics collection, and random sample generation. The executive control program of SLAM controls the simulation by advancing time and initiating calls to the appropriate event subroutines at the proper points in simulated time. Hence, the modeler is completely relieved of the task of sequencing events to occur chronologically.

A continuous model is coded in SLAM by specifying the differential or difference equations which describe the dynamic behavior of the state variables. These equations are coded by the modeler in FORTRAN by employing a set of special SLAM defined storage arrays. The value of the Ith state variable is maintained as variable SS(I) and the derivative of the Ith state variable, when required, is maintained as the variable DD(I). The immediate past values for state variable I and its derivative are maintained as SSL(I) and DDL(I), respectively. When differential equations are included in the continuous model, they are automatically integrated by SLAM to calculate the values of the state variables within an accuracy prescribed by the modeler. The event and continuous aspects of SLAM are based on GASP IV concepts (3).

An important aspect of SLAM is that alternate world views can be combined within the same simulation model. There are six specific interactions which can take place between the network, discrete event, and continuous world views of SLAM:

1. Entities in the network model can initiate the occurrence of discrete events.
2. Events can alter the flow of entities in the network model.
3. Entities in the network model can cause instantaneous changes to values of the state variables.
4. State variables reaching prescribed threshold values can initiate entities in the network model.
5. Events can cause instantaneous changes to the values of state variables.
6. State variables reaching prescribed threshold values can initiate events.

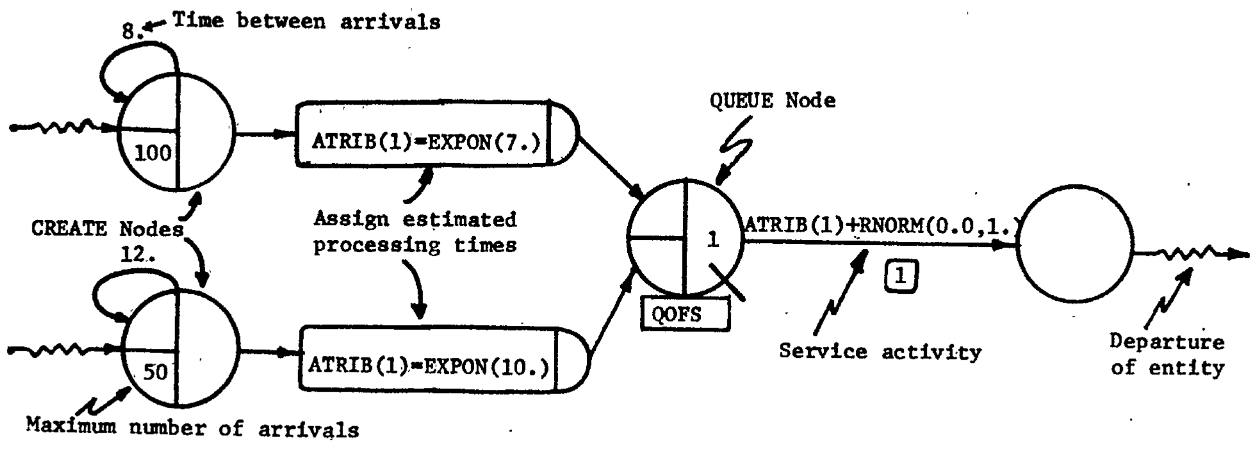
The ability to construct combined network-event-continuous models with interactions between each orientation greatly enhances the modeling power of the systems analyst. In the following sections of this tutorial, illustrations of network, event, and continuous models are given.

## A NETWORK MODEL

Consider a situation involving two types of jobs that require processing by the same server. The

FIGURE 1

Network model of a multiple entity, single server queueing situation



job types are assumed to form a single queue before the server. The network model of this situation is shown in Figure 1. The input statements corresponding to the network shown in Figure 1 are listed below.

```
NETWORK;
      CREATE,8,,,100;
      ASSIGN,ATRIB(1)=EXPON(7.);
        ACTIVITY,,,QOFS;
      CREATE,12,,,50;
      ASSIGN,ATRIB(1)=EXPON(10.);
 QOFS QUEUE(1);
        ACTIVITY/1,ATRIB(1)+RNORM(0.0,1.0);
      TERM;
      ENDNETWORK;
```

In this model, one type of entity is scheduled to arrive every 8 time units and only 100 of them are to be created. These entities have a service time estimated to be a sample from an exponential distribution with a mean time of 7. This service time is assigned to attribute 1 at an ASSIGN node. For the other type of entity, the time between arrivals is 12 time units and a maximum of 50 of these entities can be created. The estimated service time for each of these entities is exponentially distributed with a mean time of 10. Both types of entities are routed to a QUEUE node whose label is QOFS.

The server of the system is modeled as activity 1 where the service time is specified as attribute 1 plus a sample from a normal distribution. Thus, the actual processing time is equal to the estimated processing time plus an error term that is assumed to be normally distributed. This model might be used to represent a job shop in which jobs are performed in the order specified by the ranking rule specified for the QUEUE node.

## AN EVENT MODEL

In this section, we illustrate the building of a discrete event simulation model by describing the coding of a single server queueing situation. In the coding which follows, we assume that the time between arrivals is given by the exponential distribution with a mean of 20 minutes and that the service time is uniformly distributed between 10 and 25 minutes. The operation of the system is to be simulated for a period of 480 minutes.

To construct a discrete event simulation model in SLAM, the user must do the following:

1. Write a main program to dimension a filing array, specify values for input and output devices, and call SLAM;
2. Write subroutine EVENT(I) to map the user assigned event codes onto a call to the appropriate event subroutine;
3. Write subroutine INTLC to initialize user defined variables and to schedule the first arrival to the system;
4. Write event subroutines to model the logic for the arrival event and the end-of-service event; and
5. Prepare the input statements required by the problem.

In this discussion, we will concentrate on item 4. We will code the logic for the arrival event in subroutine ARVL and assign it event code number 1. The code for the logic for the end-of-service event will be written in subroutine ENDSV and it will be referenced as event code number 2.

The logic for the arrival event, ARVL, is presented in Figure 2. The values of the SLAM discrete event variables are passed to the event routine through

COMMON block SCOM1. The SLAM variable XX(1) is equivalenced to the user defined variable BUSY. The first function performed by the event is the rescheduling of the next arrival event to occur at the current time plus a sample from an exponential distribution with mean of 20.0 and using random stream number 1. The first attribute of the current entity is then set equal to the arrival time, TNOW. A test is then made on the variable BUSY to determine the current status of the server. If BUSY is equal to 0.0, then the server is idle and a branch is made to statement 10 where BUSY is set to 1.0 to indicate that the server is busy and the end-of-service event is scheduled to occur at time TNOW plus a sample from a uniform distribution between 10.0 and 25.0 using random stream number 1. Otherwise, the entity is placed in file 1 to wait for the server. In either case, the entity is identified by its arrival time which is stored as attribute 1.

FIGURE 2

Subroutine ARVL for bank teller problem.

```
   SUBROUTINE ARVL
   COMMON/SCOM1/ ATRIB(100),···
   EQUIVALENCE (XX(1),BUSY)
   CALL SCHDL(1,EXPON(20.,1),ATRIB)
   ATRIB(1)=TNOW
   IF(BUSY.EQ.0.) GO TO 10
   CALL FILEM(1,ATRIB)
   RETURN
10 BUSY=1.
   CALL SCHDL(2,UNFRM(10.,25.,1),ATRIB)
   RETURN
   END
```

The logic for the end-of-service event, ENDSV, is depicted in Figure 3. The variable TSYS is set equal to the current time, TNOW, minus the first attribute of the current entity being processed. When an event is removed from the event calendar, the ATRIB buffer array is assigned the attribute values that were associated with the event when it was scheduled. Since the value of ATRIB(1) is the entity's arrival time, the value of TSYS represents the elapsed time between the arrival and end-of-service event for this entity. A call is then made to subroutine COLCT to collect statistics on the value of TSYS as collect variable number 1. A test is made on the SLAM function NNQ(1) representing the number of entities waiting for service in file 1.

FIGURE 3

Subroutine ENDSV for bank teller problem.

```
   SUBROUTINE ENDSV
   COMMON/SCOM1/ATRIB(100),···
   EQUIVALENCE (XX(1),BUSY)
   TSYS=TNOW-ATRIB(1)
   CALL COLCT(TSYS,1)
   IF(NNQ(1).GT.0) GO TO 10
   BUSY=0.
   RETURN
10 CALL RMOVE(1,1,ATRIB)
   CALL SCHDL(2,UNFRM(10.,25.,1),ATRIB)
   RETURN
   END
```

If the number of entities waiting is greater than zero, a transfer is made to statement 10 where the first entity waiting is removed from file 1 and placed onto the event calendar. The end-of-service event is scheduled to occur at time TNOW plus the service time. If no entity is waiting, the status of the server is changed to idle by setting the variable BUSY to 0.

The input statements for this example are shown in Figure 4. The GEN statement specifies the analyst's name, project title, date, and number of runs. The LIMITS statement specifies that the model employs 1 file, the maximum number of attributes is 1, and the maximum number of simultaneous entries in the system is 20. The STAT statement specifies that collect variable number 1 is to be displayed on the standard SLAM summary report with the label TIME IN SYSTEM and that a histogram is to be generated with 10 interior cells, the upper limit of the first cell is to be 0, and the cell width of each interior cell is to be 4. The TIMST statement causes time-persistent statistics to be automatically maintained on the SLAM variable XX(1) and the results to be displayed using the label UTILIZATION. The INIT statement specifies that the beginning time of the simulation is time 0 and that the ending time is time 480. The FIN statement denotes the end to all SLAM input statements. This completes the description of the discrete event model of the single server queueing situation.

FIGURE 4

Data statements for bank teller problem.

```
GEN,C. D. PEGDEN,BANK TELLER,11/20/77,1;
LIMITS,1,1,20;
STAT,1,TIME IN SYSTEM,10/0/4;
TIMST,XX(1),UTILIZATION;
INIT,0,480;
FIN;
```

## NETWORK AND DISCRETE EVENT INTERFACE

In SLAM, a discrete event can be caused by an entity arrival to an EVENT node. When this occurs, subroutine EVENT is called with an event code established as part of the EVENT node description. In addition, attribute assignments and activity durations can be prescribed as user functions which indicates that a FORTRAN subprogram is to be written by the modeler.

An entity can be inserted into a network model from a discrete event by calling subroutine ENTER(I). Such a call causes an entity with attribute values as defined by the vector ATRIB(·) to arrive at ENTER node I. In addition, activities in a network can be stopped within an event subroutine by calls to subroutine STOPA(NTC) where NTC is a value associated with an activity duration specification. This is one of the procedures in SLAM that supports activity scanning modeling concepts.

## A CONTINUOUS MODEL

To illustrate a model of a continuous system using

SLAM, we present a model of Cedar Bog Lake that was developed by Williams (4).

The model includes three species, a solar energy supply ($x_s$), and the organic matter that forms a sediment on the lake bottom ($x_o$). These lake variables are modeled in terms of their energy content (calories/centimeter$^2$) and the energy transfers between the various lake variables and losses to the environment ($x_e$). The three species are plants ($x_p$), herbivores ($x_h$), and carnivores ($x_c$). The differential equations relating these species to the sediment and the solar energy source are shown below.

$$\frac{dx_p}{dt} = x_s - 4.03x_p \ .$$

$$\frac{dx_h}{dt} = 0.48x_p - 17.87x_h \ .$$

$$\frac{dx_c}{dt} = 4.85x_h - 4.65x_c \ .$$

$$\frac{dx_o}{dt} = 2.55x_p + 6.12x_h + 1.95x_c \ .$$

$$\frac{dx_e}{dt} = 1.00x_p + 6.90x_h + 2.70x_c \ .$$

The values of the variables at time zero are: $x_p(0) = 0.83$, $x_h(0) = 0.003$, $x_c(0) = 0.0001$, $x_o(0) = 0.0$, and $x_e(0) = 0.0$.

The annual cycle in solar radiation is simulated using the following equation:

$$x_s = 95.9(1+0.635 \sin 2\pi t)$$

where t is time in years. These equations represent such processes as the predation of one species by another, plant photosynthesis, and the decaying of dead species. Energy transfers between lake entities and their environment are due to respiration and migration.

We will use SLAM to illustrate the procedure for obtaining the values of the variables $x_p$, $x_h$, $x_c$, $x_o$, $x_c$, $x_s$ over time. First, we make an equivalence between the model variables and the SLAM state vector SS(·) as shown below.

$$SS(1) = x_p \rightarrow DD(1) = \frac{dx_p}{dt}$$

$$SS(2) = x_h \rightarrow DD(2) = \frac{dx_h}{dt}$$

$$SS(3) = x_c \rightarrow DD(3) = \frac{dx_c}{dt}$$

$$SS(4) = x_o \rightarrow DD(4) = \frac{dx_o}{dt}$$

$$SS(5) = x_c \rightarrow DD(5) = \frac{dx_c}{dt}$$

and $SS(6) = x_s$ .

The entire SLAM program consists of writing the main program, subroutine STATE, and the input statements. These are shown in Figure 5. The main program is in the standard SLAM form. In subroutine STATE, the set of differential equations is coded. The translation of the equations from the model to the SLAM code is direct and normally does not require an excessive amount of work. The input statements for the model involve mainly the definitions of the variables to be plotted which is done on RECORD and VAR input statements. The CONTINUOUS statement defines the limits on the step size and the accuracy requirements for the numerical integration of the differential equations.

FIGURE 5

SLAM program of Cedar Bog Lake.

```
 PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,
1TAPE7)
 DIMENSION NSET(1000)
 COMMON/SCOM1/ ATRIB(100),DD(100),...
1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),...
 COMMON QSET(1000)
 EQUIVALENCE (NSET(1),QSET(1))
 NNSET=1000
 NCRDR=5
 NPRNT=6
 NTAPE=7
 CALL SLAM
 STOP
 END

 SUBROUTINE STATE
 COMMON/SCOM1/ ATRIB(100),DD(100),...
1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),...
 DATAPI/3.14159/
 SS(6)=95.9*(1.+0.635*SIN(2.*PI*TNOW))
 DD(1)=SS(6)-4.03*SS(1)
 DD(2)=0.48*SS(1)-17.87*SS(2)
 DD(3)=4.85*SS(2)-4.65*SS(3)
 DD(4)=2.55*SS(1)+6.12*SS(2)+1.95*SS(3)
 DD(5)=SS(1)+6.9*SS(2)+2.7*SS(3)
 RETURN
 END

 GEN,PRITSKER,CEDAR BOG LAKE,3/5/1978,1;
 CONTINUOUS,5,1,.00025,.025,.025;
 INTLC,SS(1)=.83,SS(2)=.003,SS(3)=.0001;
 INTLC,SS(4)=0.0,SS(5)=0.0;
 RECORD,TNOW,TIME,0,P,0.025;
 VAR,SS(1),P,PLANTS;
 VAR,SS(2),H,HERBIVORES;
 VAR,SS(3),C,CARNIVORES:
 VAR,SS(4),O,ORGANIC;
 VAR,SS(5),E,ENVIRONMENT;
 VAR,SS(6),S,SOLAR ENERGY;
 INITIALIZE,0,2.0;
 FIN;
```

The INTLC statement initializes the SS(·) values as prescribed by the problem statement, and the INITIALIZE statement specifies that the simulation should start at time zero and end at time 2. This example illustrates the ease of coding continuous models in SLAM.

## DISCRETE AND CONTINUOUS INTERFACE

Values of state variables and their derivatives can be changed at ASSIGN nodes by directly replacing the values of SS(·) and DD(·). Such changes can also be performed in event subroutines. Equations in subroutine STATE can be written as a function of XX(·) values which can also be changed at ASSIGN nodes or event routines. The detection of state variables crossing thresholds can be modeled using DETECT nodes and an activity can be specified to end upon the release of a DETECT node. Input statements are provided that cause an event to occur when a state variable crosses a threshold. As can be seen, there are diverse procedures in SLAM to model interactions between network, event, and continuous model segments.

## SUMMARY

SLAM supports the modeling of systems from diverse viewpoints. It is written in ANSI standard FORTRAN and, hence, is transportable. Flexible freeform input procedures and standard summary reports have been designed into SLAM. In addition, SLAM provides all the support subprograms required by simulation modelers and analysts.

## REFERENCES

1. Pritsker, A.A.B. and C.D. Pegden, Jr., Introduction to Simulation and SLAM, Systems Publishing Corp., 1978.
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. Williams, R.B., "Computer Simulation of Energy Flow in Cedar Bog Lake, Minnesota Based on the Classical Studies of Lindeman," in Systems Analysis and Simulation in Ecology, B. C. Patten, Ed., New York: Academic Press, 1971.