SIMULATION MODELING WORKSHOP

William E. Biles
University of Notre Dame

ABSTRACT

The crucial phase in simulating a real-world system is the formulation and development of a credible model of that system. This workshop examines selected phases of model development, including (1) development of flow charts describing the mathematical-logical operation of the system, (2) the translation of this mathematical-logical model into a computer program, and (3) the verification and validation of the computer model. These concepts are illustrated through an example involving sampling inspection of roller bearing assemblies. Practical exercises will be conducted. This workshop is tutorial in nature and emphasizes the fundamentals of simulation modeling.

INTRODUCTION

Simulation is a problem-solving procedure for defining and analyzing a model of a system. Simulation can take several forms, including electrical analog, fluid analog, and the more familiar digital computer simulation. In the latter context, simulation can be defined as the establishment of a mathematical-logical model of a system and the experimental manipulation of that model on a digital computer.

This workshop concentrates on the model development phase of computer simulation. It is assumed that the problem has been defined, the significant system variables identified, and data collected and statistically analyzed. It is likewise assumed that the procedures for experimenting with a credible simulation model, including designing computer simulation experiments, analyzing simulation output, and employing appropriate optimization techniques, are sufficiently well understood. This workshop examines the sequence of steps in the model development phase, including (1) the development of program flow charts describing the mathematical-logical operation of the system, (2) the translation of this mathematical-logical model into a computer program, and (3) the verification and validation of the computer model.

Two example problems serve as illustrated of the principles and techniques discussed here. Both examples involve the sampling inspection of machined roller-bearing assemblies. The first example considers only the sampling process itself, and represents a typical Monte Carlo simulation. The second example extends the first, considering the sequence of time-events involved in the sampling inspection, and represents a typical discrete-event simulation. The workshop participants will perform these simulations by hand using tables of uniformly distributed random numbers. They will gain practice in the development of program flow charts and maintaining statistics on the progress of the simulation.

SYSTEMS AND MODELS

Computer simulation offers a convenient means of studying the behavior of a system. By system, we mean some circumscribed sector of reality upon which we focus analysis for the purpose of accomplishing some logical end. For example, we might focus on a section of a manufacturing plant where two machines feed in-process parts onto an overhead monorail conveyor which in turn feeds four other machines. There might be many other machines and conveyors in this particular plant, but we wish to focus on this section of the overall operation. Thus these six machines and the monorail conveyor connecting them, together with the in-process parts and any operators involved in this section, constitute the system which we choose to investigate.

A system is a collection of related entities, each characterized by a set of attributes. These entities engage in activities, which elapse over time and culminate in events. An activity may last some known or deterministic time, or some unknown or probabilistic time. An event, which occurs at an instant in time, marks the termination of an activity and alters the state of the system by changing the values of the attributes associated with one or more entities. For example, in the machining operation cited earlier, the entities in the system are the six machines, the buckets on the monorail conveyor, the in-process parts in the system, and the operators running the system. The attributes of a machine might be its operational state (busy, failed, blocked on the output side, blocked on the input side), the number of parts it can service at one time, and the rate at which it processes parts. Attributes of a bucket on the monorail conveyor might be its state (empty or loaded) and its position on the monorail loop. An activity in which a machine engages is the machining operation, which is also an activity for each of the parts being machined. An activity associated with a monorail bucket is its movement between machine positions.

In simulation modeling, we seek to (1) identify the entities in the system, (2) characterize these entities in terms of their attributes, (3) determine the activities in which the entities engage, (4) establish the various states the system can have, and (5) develop the mechanisms which relate these several facets of the system to one another. We collect and analyze data, which allows us to provide the necessary input to the model, operate the model (simulate), and evaluate the output from the model. We develop a flow chart which reflects the logical relationships involved in the system being modeled. We translate the flow chart and the results of an analysis of our data into a computer program. For this purpose we may use one of the general purpose programming languages (FORTRAN, ALGOL, BASIC, PL/1), a simulation language (GPSS, SIMSCRIPT, GASP-N), or a specialized simulation modeling technique (GERTS). The choice of the language or technique to be used is usually dictated by such considerations as availability and familiarity.

DEVELOPING A FLOW CHART

It is well nigh impossible to describe the process of developing a flow chart without doing one straight away. To illustrate the development of a flow chart to initiate simulation model development, let us consider a simple Monte Carlo example. That is, there are random processes in the model, but no time-sequencing of events.

Precision Bearing Company wishes to evaluate a new sampling inspection procedure. For each bearing assembly produced, one of three levels of "inspection" can occur.

Inspection	Percent	Percent Rejected
1. No Inspection	75%	0%
2. Casual Inspection	20%	2%
3. Detailed Inspection	5%	5%

Finished assemblies are randomly shuttled into one of these three inspection modes with the given probabilities. For a given inspection procedure, the rejection rates (%) are as stated.

First, let us identify the essential entities in this simple system. The bearing assemblies and the inspectors are the two classes of entities we must consider. The system state would simply be the total numbers of bearing assemblies accepted and rejected at any point in the simulation.

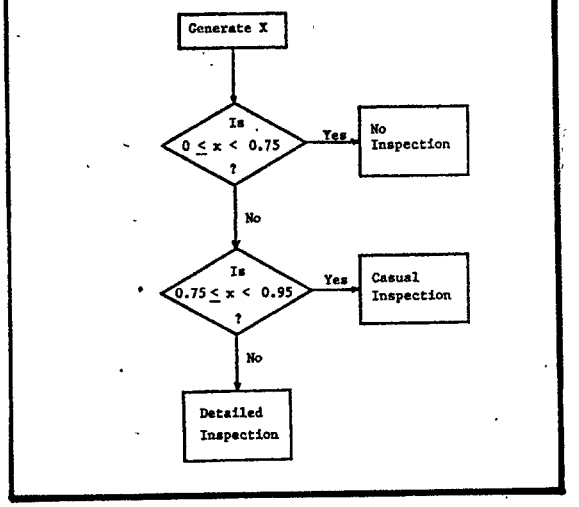
How do we cause a given bearing assembly to undergo a particular inspection mode? We sample from a uniform distribution in the interval $0 \leq x \leq 1$. We then choose the inspection mode as follows:

$0 \leq x < 0.75$	No Inspection
$0.75 \leq x < 0.95$	Casual Inspection
$0.95 \leq x < 1.00$	Detailed Inspection

The flow chart corresponding to these logical steps is as shown in Illustration 1 below.

ILLUSTRATION 1

Choosing a Mode of Inspection



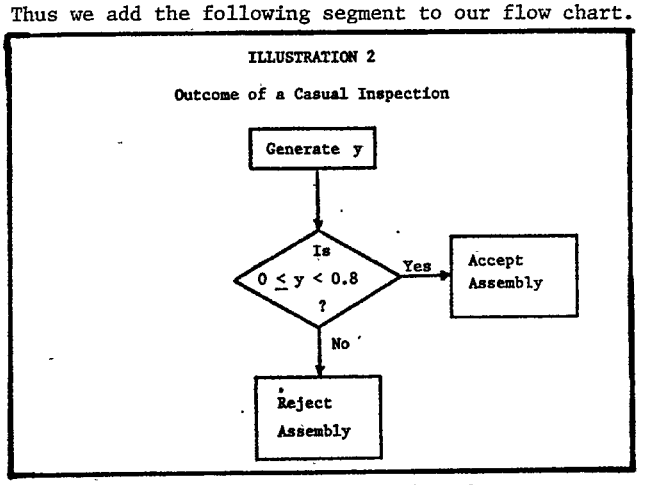
We employ a similar process for determining the outcome of each inspection. For the casual inspection, we sample from a uniform distribution in the interval $0 \leq y \leq 1$. We generate the inspection outcome as follows:

$0 \leq y < 0.98$	Accept Bearing Assembly
$0.98 \leq y < 1.0$	Reject Bearing Assembly

Thus we add the following segment to our flow chart.

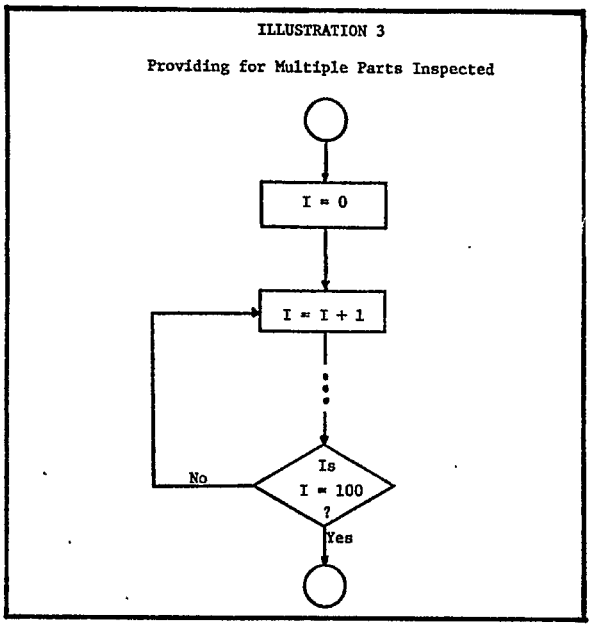
ILLUSTRATION 2

Outcome of a Casual Inspection



In this example we see that the flow chart represents the sampling inspection procedure applied to one bearing assembly. If we wish to perform 100 such simulated inspections, we would add the following components to our flow chart.

Suppose that we now wish to add to our model the capability for generating the time events associated with this sampling inspection procedure. There are basically two such time events: (1) an arrival of a finished assembly at the inspection station, and (2) the end of inspection on the assembly. Suppose



that the arrival rate is Poisson distributed with mean rate λ units per hour. Therefore the time between arrivals is exponentially distributed with mean time $1/\lambda$ hours. We can use the occurrence of one arrival to schedule the next arrival. The relationship which accomplishes this event scheduling mechanism is

$$t_{next} = t_{now} - (1/\lambda) \ln z$$

where z is a uniformly distributed random variable $0 \leq z \leq 1$. The term $\ln z$ is a negative quantity, and hence $[-(1/\lambda) \ln z]$ is a deviate $0 \leq \Delta t \leq \infty$ which has mean $(1/\lambda)$. The flow chart segment needed to illustrate these steps is as shown in Illustration 4. Of course, we typically start our simulation at $t_{now} = 0$.

Suppose the inspection process is uniformly distributed $0.1 \leq w \leq 0.2$ hours for the detailed inspection. When a new arrival has taken place and the sampling inspection calls for a detailed inspection, the following relationship applies:

$$t_{next} = t_{now} + (0.1 + 0.1r)$$

where $0 \leq r \leq 1$ is a uniformly distributed random number.

The management of the queue of parts available for inspection is easily shown in a flow chart, but rather tedious to program. To accomplish this task, we must establish a file containing the parts waiting for inspection. The entries in this file are waiting parts; that is, entities. Each entity (part) is characterized by two attributes: (1) the time it arrived, and (2) the type of inspection to be performed. If we have one inspector performing all inspections on a FIFO basis, we can maintain the file in order of arrivals, saving the arrival time for possibly maintaining statistics on total time in the inspection station.

PREPARING THE SIMULATION PROGRAM

Once the flow chart has been fully developed, it is necessary to translate the model into a computer program. The flow chart provides the mechanism by which the sequence of program statements and control are crafted. For example, in FORTRAN we generate uniformly distributed random numbers (actually "psuedo-random numbers" because of the algorithm employed) by calling a subroutine. The statements

```
      I = 19743
          .
          .
          .
      CALL RANDU(I,J,X)

      I = J
```

generate a uniformly distributed value $0 \leq x \leq 1$.

The test of x to determine which type of inspection is invoked might consist of the logical IF statements

```
      IF(X.LT.0.75) GO TO 50
      IF(X.LT.0.95) GO TO 25
```

At statement 50 the mechanics of the "no inspection" process would be coded, while at statement 25 the casual inspection" is performed. Immediately following the second IF statement would be placed the mechanism for the "detailed inspection", which might consist of the statements

```
      IF(Y.LT.0.95) GO TO 15
      WRITE (6,10)
   10 FORMAT (5X,'PART ACCEPTED - DETAIL')
      GO TO 5
   15 WRITE (6,20)
   20 FORMAT (5X, 'PART REJECTED - DETAIL')
      GO TO 5
```

where statement 5 generates a new arrival.

VERIFICATION AND VALIDATION

The process of establishing the credibility of the computer simulation model, and hence its suitability as an instrument in engineering design or economic decision - making, involves two separate activities; (1) verification and (2) validation. Verification is the activity in which we ascertain that the computer program performs as intended. Validation is the process by which we ensure that the model behaves like the actual system. These two activities are basically performed sequentially, although program changes made during the validation phase will require verification.

The process of verification involves "debugging" as well as perfecting logical operations. For instance, in the example cited earlier involving the interaction between a monorail conveyor and six machines we would verify that, when an empty bucket coincides with a machine having a part in the output position, the bucket does indeed pick up the part. Otherwise (that is, a full bucket or an empty pick-up station) no pick-up is executed. We could observe that the computer program executes, but through faulty logic "pick-ups" are made when they shouldn't and vice versa. Thus verification involves program debugging and careful checking of program logic.

Validation requires us to execute the computer simulation model at model input values that correspond to known conditions in the actual system. The model output values are then compared to analogous values for the real system. Statistical techniques are employed to test the hypothesis that the model behaves like the actual system. Done properly, validation consists of completely objective evaluations. When the validation process is completed, we can then perform designed experiments, employing the simulation model as the experimental environment, and optimize the design in terms of projected system performance. Working with a model whose credibility has been so confirmed, we can confidently make decisions in terms of the actual system (realizing, of course, that there is some small likelihood of being wrong).

SUMMARY

This workshop has reviewed the fundamental concepts in developing a credible computer simulation model. Several other sessions and workshops at this conference address other areas, such as random number and random variate generation, simulation languages, and statistical methodology in simulation. Ideally, someone who has just become acquainted with discrete-event simulation would find these tutorial sessions and workshops extremely useful in gaining the background necessary to begin applying simulation to the engineering and economic problems confronting his or her work environment.