Discrete-Event Simulation

Prateek Sharma

Abstract: Simulation can be regarded as the emulation of the behavior of a real-world system over an interval of time. The process of simulation relies upon the generation of the history of a system and then analyzing that history to predict the outcome and improve the working of real systems. Simulations can be of various kinds but the topic of interest here is one of the most important kind of simulation which is Discrete-Event Simulation which models the system as a discrete sequence of events in time. So, this paper aims at introducing about Discrete-Event Simulation and analyzing how it is beneficial to the real world systems.

Index Terms: Simulation, Discrete-event, Steps in a simulation study, Simulation models, Single-Server queue, Advantages & disadvantages of simulation, Applications of simulation.

1 INTRODUCTION

Simulations are of great importance as they prevent the catastrophic failures in the system due to impact of a change. New changes, procedures, information flows etc. can be examined without interrupting the smooth working of real systems. A simulation model is developed to study the working of a system as it evolves over time. A fully developed and validated model can answer a variety of questions about real systems. Of all the simulation techniques, Discrete-Event Simulation is the one which models the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change is assumed to occur; thus the simulation can directly jump in time from one event to the next.

1.1 STEPS OF SIMULATION

Problem formulation: Before making an attempt to solve a problem, we need to first formulate the problem so that it is clearly understood. It is important that all the stakeholders understand and agree with the problem statement. Often, ambiguous problem statements prove a bottleneck in the development of the simulation model. Set objectives and plan: After the problem statement is clearly formulated, objectives have to be established. Objectives are important because they specify the questions to be answered by the simulation. And depending upon the objectives, a decision has to be reached if the simulation is an appropriate technique for the problem at hand. Once the decision for simulation is taken, an overall project plan should be prepared which should indicate the amount of resources required, no. of phases into which the project will be divided, deadline for each phase and output at the end of each phase. Conceptualize model: This is the most important and arduous step. Conceptual modelling is related to abstraction. Abstraction emphasizes the need for simplification of the real system and for assumptions about what is unknown about the real systems. Conceptual modelling is the abstraction of the simulation model of a real system. More formally, conceptual modelling can be described as an elucidation of the computer simulation model, not specific to any software, describing the objectives, inputs, outputs, content, assumptions and simplifications of the model. However, model complexity should not exceed that required to fulfill the purpose. One-to-One mapping between the model and the real system is necessary for a good conceptual model. Data collection: Simulation projects count on high input data quality. Thus, data collection is very important and consumes an extensive amount of time. As the complexity of the model changes, the required data elements can also change. Model translation: This step consists of translating the conceptual model into a computer-recognizable format. It is possible that the result can be achieved with little or no coding depending on the model. Usually, the modeler can program the model using a simulation programming language or use a special-purpose simulation software. Simulation softwares can drastically reduce the effort but it is not always that the problem is amenable to a solution with any simulation software. Verification & Validation: Verification is carried out to ensure if the program developed during model translation is according to the requirements and design specifications. Verification requires a great deal of debugging. Verification can be deemed acceptable when the logical structure of the model is correctly represented in the computer. Validation is carried out to ensure that the program actually meets the simulation's needs and that the assumptions were correct in the first place. Validation is done by comparing the model against actual system and then improving the model by using the discrepancies. This process is repeated until the model accuracy is deemed acceptable. So, in short, verification answers to the question of "Are we developing the product right?" and validation answers to the question of "Are we developing the right product?" Experimental design: Simulation models often have many input factors and determining which ones have a significant impact on performance measures of interest can be a daunting task. This step is used to determine which factors have the greatest effect on responses. Documentation & reporting: A final report containing the result of all the analysis clearly and succinctly can prove a great deal of help to model users in reviewing the final formulation and comparing the results of the experiments. Both, program and progress should be clearly documented. The documentation can act as a reference guide for all the future problems of similar kind. The common phrase, 'If it's not in writing, it didn't happen', signifies the importance of documentation and reporting.

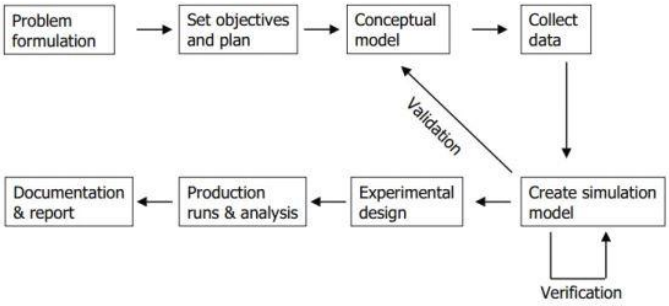

Fig. 1. *Steps in a simulation study.*

2 SIMULATION MODELS

Sometimes, it is required to experiment with the operation of real system to study the effects of the change which can lead to more efficient and advance system than the current one. But it is not a wise idea to implement the change directly in the real system as it can cause unexpected results which ultimately disrupt the working of the system. For example, in case of a bank, reducing the numbers of tellers to study the effect on the length of waiting lines might lead to the customers moving their accounts to competitors. Consequently, the task of experimenting with the real system is achieved with the help of a model of the system. A model is defined as a representation of a system for the purpose of studying the system. The model should be built so as to permit valid conclusions to be drawn about the real system. Sometimes, different models of the same system are required to be built to study different aspects of the real system. A discrete-event simulation model is both stochastic and dynamic with the special property that the changes occur at discrete times only. A stochastic simulation model has one or more random variable as inputs. Random inputs lead to random outputs. Since the outputs are random, they can be treated only as approximations of the true characteristics of a model. The simulation of a bank would involve random interarrival times and random service times. A special feature of discrete-event simulation is that random components can be taken into account without a dramatic increase in the complexity of the system model at the computational level. Dynamic simulation models represent systems as they change over time. Simulation of a bank from 9:00 A.M. to 4:00 P.M. is an example of a dynamic simulation. The passage of time plays a significant role in dynamic models. In discrete-event simulation model, the state of the system is a piecewise-constant function of time. For example, in OS scheduler, the number of jobs in a queuing system is a natural state variable that only change value at those discrete times when a job arrives or departs. Following the conventions of the process view of simulation, the models will contain active entities, which correspond to the active parts of the system being modeled, and passive entities, which correspond to the resources, queues, and other non active parts of that system. In discrete event simulation models, there are two kinds of users:

Developers: these are the users who design and build the simulation model and then verify it.

Experimenters: these are the users who observe the unraveling of the events of a model and collect statistics about the result.

2.1 WORKING OF A SIMULATION MODEL

As mentioned earlier, the process view divides a modeled system into active entities and passive entities. The active entities carry out activities, such as workers in a factory. Passive entities are not active themselves but affect the behavior of active entities in significant ways. Typical passive entities are resources, semaphores and buffers. The ability of an active entity to carry out some activity often depends on whether a certain amount of some resource is currently available. If the active entity requests an amount that is available, that active entity moves to a state indicating that it is performing the activity, and that resource moves to a state where less resource is available for succeeding requests. If the required amount is not available then the active entity will have to wait until some other active entity has finished. At any point in the simulation, an active entity is in one of the three states: Active, where it is responding to an event in its lifecycle; only one active entity can be in this state at a time and its event time defines the current simulated time. Blocked, waiting for a request to be satisfied by a passive entity. Waiting, for simulated time to reach this object's next event time. Any simulated event should cause a message to be sent to a trace file, so that an experimenter can follow the detailed internal behavior of the model. Statistics is collected by updating information about passive entities and about other values for which information is needed. The conditions under which a model executes are varied to observe how the system would respond. The below Activity diagram shows the overall flow of control.

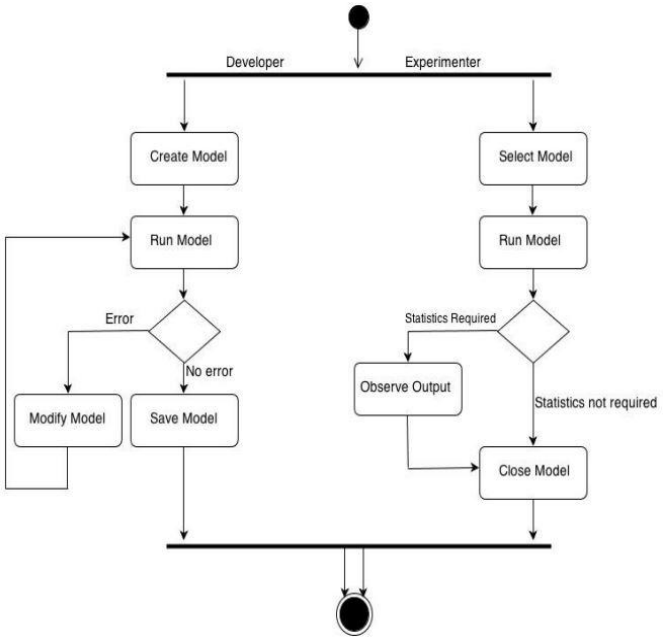

Fig. 2. *Activity diagram showing workflows of stepwise activities and actions with support for choice, iteration and concurrency. There are different set of activities for the two users, Developer and Experimenter, and both follow a sequence of activities which are clearly depicted.*

3 SINGLE-SERVER QUEUE

One of the classic examples of discrete-event simulation is a Single-Server Queue.

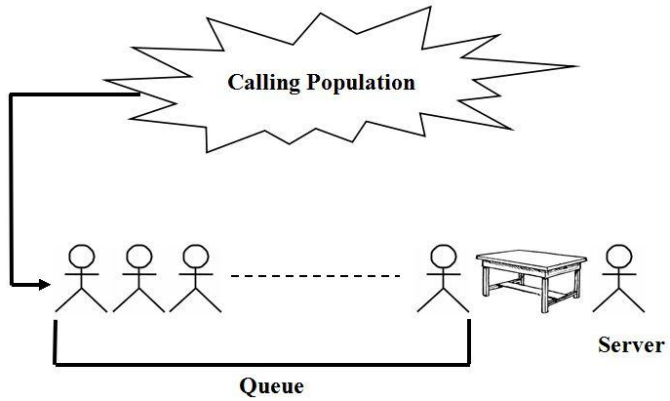

Fig. 3. *Representation of a Single-Server Queue*

In a Single-Server Queue, the Calling Population is infinite; that is, if a unit leaves the Calling Population and joins the waiting line or enters service, there is no change in the arrival rate of other units that could need service. Arrivals for service occur one at a time in a random fashion. Service times are some random length according to a probability distribution which does not change over time. The system capacity has no limit, meaning that any number of units can wait in line. Finally, units are served in the order of their arrival by a single server. Jobs (customers) arrive at the single server at random points in time in seek of service. When service is provided, the service time involved is also random. At the completion of service, jobs depart. The server operates as follows: as each new job arrives, if the server is busy then the job enters the queue, else the job immediately enters service; as each old job departs, if the queue is empty then the server becomes idle, else a job is selected from the queue to immediately enter service. At any time, the state of the server will either be busy or idle and state of the queue will be either empty or not empty. The Single-Server Queue model is built with the intention of answering the following relevant questions:

What is the mean waiting time?
What is the mean queue length?
What is the mean length of a busy period?
How does the performance change if we speed up the server?

Control of the queue is determined by the queue discipline i.e. the algorithm used when a job is selected from the queue to enter service. The standard algorithms are:
FIFO : first in, first out
LIFO : last in, first out
SIRO : service in random order
Priority : typically, shortest job first (SJF)

Certainly, the most common queue discipline is FIFO which is also known as FCFS (first come, first served). FIFO ensures that the order of arrival at the server and the order of departure from the server are the same. This observation leads to the simplification of the simulation model. There are two additional important assumptions. First, service is non-preemptive i.e. after initiation, a job will be continued to be serviced until completion. No other job can preempt the current job being serviced. Second, service is conservative i.e. the server will never sit idle if there is one or more jobs in the queue.

		Queue Status	
		Not empty	Empty
Server Status	Busy	Enter queue	Enter queue
	Idle	Impossible	Enter service

Fig. 4. *Possible job actions upon arrival*

		Queue Status	
		Not empty	Empty
Server Outcomes	Busy		Impossible
	Idle	Impossible	

Fig. 5. *Server outcomes after the completion of service*

4 SPECIFICATION MODEL

After their arrival to the service queue, jobs are indexed by i=1,2,3,… For each job there are six associated time variables:

The arrival time of job i is a_i
The delay of job i in the queue is $d_i \geq 0$.
The time that job i begins service is $b_i = a_i + d_i$.
The service time of job i is $s_i > 0$.
The wait of job i in the service system (queue and service) is $w_i = d_i + s_i$.
The time that job i completes service (the departure time) is $c_i = a_i + w_i$.
The interarrival time between jobs i-1 and i is $r_i = a_i - a_{i-1}$.

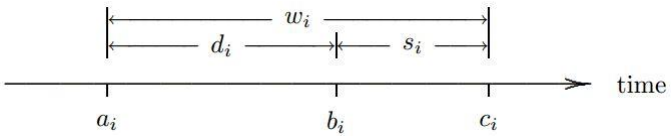

Fig. 6. *Six time variables associated with job i.*

4.1 ALGORITHMIC QUESTION

Given a knowledge of the arrival times a_1, a_2,…, the associated service times s_1, s_2,…, and the queue discipline, how can the delay times d_1, d_2,… be computed? If the queue discipline is FIFO then there is a simple algorithm for computing d_i for all i. The delay d_i of job i = 1, 2, 3,… is determined by when the job's arrival time a_i occurs relative to the departure time c_{i-1} of the previous job. There are two cases to consider. Case I. If $a_i < c_{i-1}$, i.e., if job i arrives

before job i-1 departs then job i will experience a delay of di = $c_{i-1} - a_i$.

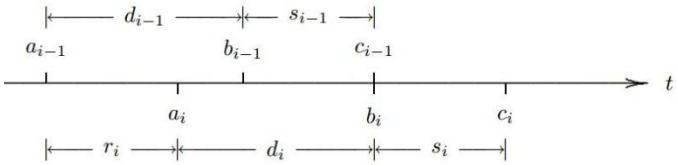

Fig. 7. *Job i arrives before job i-1 departs.*

Case II. If instead $a_i \geq c_{i-1}$, i.e., if job i arrives after (or just as) job i-1 departs then job i will experience no delay so that $d_i = 0$.

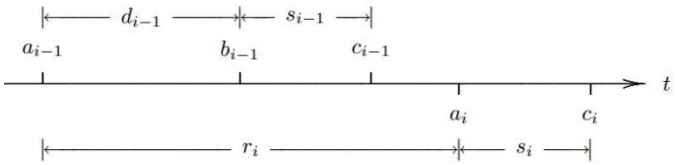

Fig. 8. *Job i arrives after job i-1 departs*

So, it is clear that the truth of the expression $a_i < c_{i-1}$ determines whether or not job i will experience a delay.

4.1.1 ALGORITHM

If the arrival times a_1, a_2, … and service times s_1, s_2, … are known and if the server is initially idle, then this algorithm computes the delays d_1, d_2, … in a single-server FIFO service with infinite capacity.

```
c0 = 0.0;                                   /* assumes that a0 = 0.0 */
i = 0;
while ( more jobs to process ) {
   i++;
   ai = GetArrival();
   if (ai < ci-1)
     di = ci-1 - ai;                        /* calculate delay for job i */
   else
     di = 0.0;                              /* job i has no delay */
   si = GetService();
   ci = ai + di + si;                       /* calculate departure time for job i */
}
n = i;
return d1, d2, ..., dn;
```

4.1.2 OUTPUT STATISTICS

One basic question that needs to be aptly answered is what statistics should be generated when constructing a discrete-event simulation. From a job's (customer's) perspective the most important statistic might be the average delay (the smaller the better), whereas from management's point of view, the server's utilization (the proportion of busy time) is the most important (the larger the better). Job-Averaged Statistics: for the first n jobs, The average interarrival time is:

$$\bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i = \frac{a_n}{n}$$

and average service time is:

$$\bar{s} = \frac{1}{n}\sum_{i=1}^{n} s_i$$

The reciprocal of the average interarrival time is the arrival rate and the reciprocal of the average service time is the service rate. Similarly, for the first n jobs, The average delay in the queue is:

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

and the average wait for the service is:

$$\bar{w} = \frac{1}{n}\sum_{i=1}^{n} w_i$$

The three statistics $\bar{w}, \bar{d}$ and $\bar{s}$ are together called as job-averaged statistics i.e. the data is averaged over all jobs. Time-Averaged Statistics: three additional variables are required to define the time-averaged statistics for a single-server queue :

l(t) = 0, 1, 2, … is the number of jobs in the whole system at time t;
q(t) = 0, 1, 2, … is the number of jobs in the queue at time t;
x(t) = 0, 1 is the number of jobs in service at time t.

By definition, l(t) = q(t) + x(t) for any t > 0.

Over the time interval (0, T) the time-averaged number in the whole system is:

$$\bar{l} = \frac{1}{\tau}\int_0^{\tau} l(t)\,dt$$

and the time-averaged number in the queue is:

$$\bar{q} = \frac{1}{\tau}\int_0^{\tau} q(t)\,dt$$

and the time-averaged number in service is:

$$\bar{x} = \frac{1}{\tau}\int_0^{\tau} x(t)\,dt$$

The three statistics $\bar{l}, \bar{q}$ and $\bar{x}$ are together called as time-averaged statistics.

5 ADVANTAGES OF SIMULATION

a) Simulation allows the study of and experimentation with a complex system.
b) It enables the feasibility testing of any hypothesis about how or why certain phenomena occur.

c) Flexibility in time handling as it can be compressed or expanded to allow for a speed-up or slow-down of the phenomena under investigation.
d) Designing of a simulation model helps in gaining knowledge that could lead to the improvement of the system.
e) Evaluating the different circumstances of simulation by changing the inputs and observing the resultant outputs can produce valuable insight into which variables are the most important.
f) New hardware designs, physical layouts, transportation systems, and so on can be tested without committing resources for their acquisition.
g) Simulation helps in formulation and verification of analytical solutions.

6 DISADVANTAGES OF SIMULATION

a) Special training is required to build simulation models. Very often, attempt to gain insights through simulation turns futile due to ambiguous models.
b) Since so much randomness is associated with simulation (random inputs) so it can be hard to distinguish whether an observation is a result of system interrelationships or of randomness.
c) Simulation modeling and analysis can be time consuming and expensive.

7 APPLICATIONS OF SIMULATION

Computer Network Simulation: simulate new protocols for different network traffic scenarios before deployment. Business Process Simulation: agent-based modeling and simulation of store performance for personalized pricing, modeling and simulation of a telephone call center, human fatigue risk simulations in continuous operations. Hospital Applications: modeling front office and patient care in ambulatory health care practices, estimating maximum capacity in an emergency room, reducing the length of stay in an emergency department. Design Implementation: to overcome implementation problems occurring in typical program evaluations like attrition problems, data coding errors, floor and ceiling effects on measures etc. that degrade the theoretical quality of these designs. Vehicle Manufacturing: to study the outcome of various factors which affect the production process like absence of labor force, undermined manufacturing times, equipment failures, lack of work or blockage etc., provide animation possibility suitable for layout design. Transportation Modes and Traffic: simulating aircraft-delay absorption, runaway schedule determination by simulation optimization, simulation of freeway merging and diverging behavior etc.

8 CONCLUSION

This paper has reviewed the flexibility provided by discrete-event simulation which can be used to model a wide variety of problems. Although it significantly facilitates the representation of complex systems but still there are a range of issues along the model development, parameter estimation, implementation and analysis that should be resolved to maximize the efficiency of the final output. Also, simulation should be avoided when the problem could be solved analytically because simulation can consume a lot of time and money. There is no doubt that simulations will be much more faster tomorrow than it is today due to the advances being made in hardware everyday permitting rapid running of scenarios. Thus, in near future, very likely discrete-event simulation is going to prove itself ubiquitous with more efficient working and accurate results.

REFERENCES

[1] Jerry Banks, John S. Carson II, Barry L. Nelson and David M. Nicol, Discrete-Event System Simulation, 2011

[2] Lawrence Leemis and Steve Park, Discrete-Event Simulation: A First Course, 2004