*Proceedings of the 2009 Winter Simulation Conference*
*M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.*

# INTRODUCTION TO SIMULATION

K. Preston White, Jr.

Department of Systems and Information Engineering
P.O. Box 700747, 151 Engineers' Way
University of Virginia
Charlottesville, VA 22904-4747, U.S.A.

Ricki G. Ingalls

School of Industrial Engineering and Management
322 Engineering North
Oklahoma State University
Stillwater, OK 74078, U.S.A.

## ABSTRACT

Simulation is experimentation with a model. The behavior of the model imitates some salient aspect of the behavior of the system under study and the user experiments with the model to infer this behavior. This general framework has proven a powerful adjunct to learning, problem solving, and design. In this tutorial, we focus principally on discrete-event simulation—its underlying concepts, structure, and application.

## 1 MODELS AND SIMULATION

A *model* is an entity that is used to represent some other entity for some defined purpose. In general, models are simplified abstractions, which embrace only the scope and level of detail needed to satisfy specific study objectives. Models are employed when investigation of the actual system is impractical or prohibitive. This might be because direct investigation is expensive, slow, disruptive, unsafe, or even illegal. Indeed, models can be used to study systems that exist only in concept.

Simulation is a particular approach to studying models, which is fundamentally experiential or experimental. In principle, simulation is much like running field tests, except that the system of interest is replaced by a physical or computational model. Simulation involves creating a model which imitates the behaviors of interest; experimenting with the model to generate observations of these behaviors; and attempting to understand, summarize, and/or generalize these behaviors. In many applications, simulation also involves testing and comparing alternative designs and validating, explaining, and supporting simulation outcomes and study recommendations.

## 2 APPLICATION DOMAINS

We might divide applications of simulation broadly into two categories. The first includes so-called *man-in-the-loop* simulations used for training and/or entertainment. Many professionals hone their skills and learn emergency procedures in simulated environments which are safe from the consequences of inexperience and failure. Pilots train in flight simulators in order to experience the cockpit of a particular aircraft; nuclear power-plant operators routinely recertify in control-room simulators; physicians learn new procedures employing simulated patients. In the realm of entertainment, we have all played computer games that simulate everything from driving a train to navigating the fanciful unrealities of virtual worlds. The emphasis here is experiential—learning (or just having fun) by doing.

The second category includes the analysis and design of artifacts and processes. This is the technical domain which engineers and operations researchers most commonly associate with simulation. Consider for example the design of a new aircraft. The Wright brothers invented the wind tunnel in order to simulate aerodynamic phenomena using scale models. Wind tunnel tests are still used to calibrate highly-complex aerodynamic computer simulations.

Simulation stands in contrast to analytical approaches to the solution of models. In an analytical approach, the model is expressed as a set of equations that describe how the system state changes over time. We solve these equations using standard mathematical methods—algebra and calculus—to determine the distribution of the state at any particular time. The result is a general, closed-form solution, which gives the state at any time as a function of the initial state, the input, and the model parameters. When models can be solved analytically this is always the preferred approach. However, for complex systems this is almost never the case.

Simulations also may be categorized according to their implementation strategy. Continuous system simulation, Monte Carlo simulation, discrete-event simulation, hybrid simulation, and agent-based simulation all have there particular implementation strategies. In keeping with the theme of this conference, we restrict this tutorial to further consideration of DES. Throughout this paper, we will be referencing ideas and thoughts from Shannon (1975), Law (2007), Banks, *et. al.* (2000), Kelton, Sadowski, and Swets (2010), Ingalls and Kasales (1999), Ingalls (1998), and Ingalls and Eckersley (1992), and (White, 2007).

## 3 CALL CENTER EXAMPLE

In order to illustrate the concepts developed in the remainder of this paper, we will refer to a simple example—the processing of phone calls at an inbound call center for a major retailer of home electronics products. To keep it simple, we limit the scope and ignore the details and complexities of actual call-center operations. (For an industrial-strength treatment, see, for example, Chung and White (2008)). In this and the following sections, we follow the presentation of ideas developed in Ingalls (2008).

The process logic for this simulation model is shown in Figure 1. Arriving calls first connect to a telephone switch. If the number of calls currently on hold is greater than ten, the caller receives a busy signal and immediately hangs up. Otherwise, the call is delivered to an automated interactive voice response (IVR) unit. The caller is asked to, "Dial one for car-stereo products; dial two for all other products" and the call is routed accordingly. The call then waits in the appropriate queue (listening to classic rock) until the first sales representative servicing the identified product type becomes available. Finally the call is processed and the caller hangs up. For midday peak periods during the upcoming Christmas season, the call-center manager would like to know the minimum number of each type of sales representative needed to insure that (i) fewer than 2% of call waiting times for either product are greater than 1 min and (ii) fewer than 3% of all incoming calls are refused at the switch.

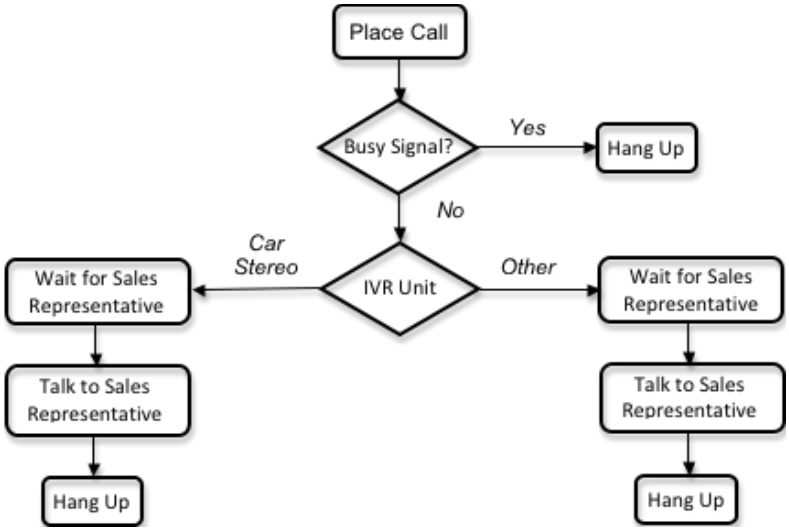


Figure 1: Call Center Example Logic Flow

## 4 DISCRETE-EVENT SIMULATION STRUCTURE

Although there are various paradigms for discrete-event simulation, a basic structure has evolved that is used by most simulation software packages. Regardless of how complex a discrete-event simulation package may be, it is likely to contain the basic structural components described in this section.

### 4.1 Inputs, Outputs, and State

The actions of the environment on a system are called the *inputs* to the system. These inputs cause changes in the internal condition of the system, called the system *state*. The *outputs* of the system are those measured quantities, which can be derived from the system state, which we need to know in order to answer the questions posed for the simulation study. In other words, the inputs cause changes in the system state and these are reflected by changes in the output.

### 4.2 Entities and Attributes

In a discrete-event simulation, inputs are realized by the arrival of dynamic *entities*. These entities flow through the system and are the structural elements which effect the changes in the *system state variables*. Without entities, nothing would happen. Indeed, one stopping condition for a simulation run is when there are no active entities in the system.

In our example, the entities are telephone calls—customers seeking information and perhaps wanting to place an order for a product. The state of the system at any point in time can be defined by three state variables—the number of calls in process on the IVR unit and the number of calls of each type waiting for service or in process. Obviously, the state changes every time a call of either type arrives or departs one of the processing units. While for convenience we will monitor a large number of variables during the simulation (the system *image*), in principle each of these variables can be derived if we know the inputs to the system and the system state at all points in time during the simulation run.

Entities have *attributes*–characteristics of a given entity that are unique to that entity. Attributes are critical to understanding the performance and function of entities in the simulation. In our example, each entity has three attributes. The first (Product_Type) is the class of product requested by the caller. We need this attribute in order to determine the routing of the call through the system. The second (Start_Time) is the time that the entity arrives and either receives a busy signal or joins the appropriate queue to begin waiting for service. The third (Begin_Wait) is the time the call leaves the IVR unit and joins one of the queues. We need these two attributes in order to determine the system time and the duration of waiting time for each entity.

In more complex simulations, there can be many different entity types. These can represent transactions (as in our example), physical objects, people, information—anything that can cause a state change. Indeed, artificial entities are used to implement control logic (such as the time the simulation ends) in the model.

### 4.3 Activities and Events

*Activities* are processes and logic in the simulation. *Events* are conditions that occur at a point in time which cause a change in the state of the system. An entity interacts with activities to create events. There are three major types of activities in a simulation: *delays, queues, and logic*.

A *delay* activity occurs when the flow of an entity is suspended for a *definite* period of time. In our example, there are two delays, each occurring while a given type of call receives service from the corresponding type of sales representative. In general, the length of time for a delay is either constant or is randomly generated. At the point that the entity starts the delay, an event occurs. This event puts the entity on a list called the *calendar* (which we will get to later). If the delay is for *d* time units, then the entity is scheduled to complete the delay *d* time units after the current time of the simulation. At that time, the delay expires and another event is generated.

A *queues* activity occurs when the flow of an entity is suspended for an *unspecified* period of time. Entities can be waiting for *resources* (which we will get to later) to be come available or for a given system condition to occur. *Queues* are most commonly used for waiting in line for a resource or storing material that will be taken out of the queue when the right conditions exist. In our example, there are two queues, each occurring while a given type of call waits for service from the corresponding type of sales representative (because all of these representatives are currently busy serving other calls). Both of these queues contain entities waiting for *resources* to become available.

*Logic* activities simply allow the entity to effect the state of the system through the manipulation of state variables or decision logic. The first of several logic activities in our example is the decision whether or not to accept and arriving call into the system. This decision is determined based on the total number of entities currently waiting in queue for service.

### 4.4 Resources

*Resources* represent anything in a simulation that has a constrained capacity. Resources are time-shared by entities, entities must queue for busy resources, and entities typically are delayed after these have seized a resource and begin processing. Common examples of resources include workers, machines, nodes in a communication network, and traffic intersections.

In our example, we have two resources, each representing the sales force for a given product type. The study question is to determine the capacity of each of these resources needed to satisfy the manager's operating policy objectives. Note that, since we assume that there is nothing individuating about the sales representatives, we represent each pool of representatives as a single resource with the potential capacity to process multiple calls simultaneously. This assumption easily could be relaxed by modeling each pool as a *set* of individuated resources with different processing capabilities.

Note also that after a call is processed, the caller hangs up and the resource is free to process any call waiting in the corresponding pool. Therefore all of the time an entity is control of a unit of this resource capacity is productive. This need not be the case in general, if an entity is blocked from releasing the resource by some downstream condition. For this reason it is common to account for both the productive and unproductive time that a resource is held.

Note finally that in our example, we assume that the switch acts instantaneously. Therefore there is no need to model the switch as a resource—it is a logical activity. We also assume that the IVR unit has sufficient capacity to handle as many calls it receives simultaneously. We need to account for the delay time associated with the IVR processing, but do not need to model the IVR as a resource. The IVR is a pure delay activity.

Very complex resources can be modeled in a simulation. In a manufacturing simulation, for example, conveyors are a complex resource type that many simulation packages offer. Also, transportation options such as trucks are offered as resources. A third complex resource is a vat or container that has a (continuous) flow of material both in an out of the resource. Depending on the target market of the simulation package, many complex resources are available to use

### 4.5 Global Variables

If you are a programmer, then the idea of having *global variables* is nothing new. The values of a *global variable* are available to the entire simulation at all times and can track just about anything of interest. In our model we have eight global variables. One variable helps to configure the problem. This is the limit on the number of calls on hold permissible (Max_On_Hold). While this value is given as ten in the problem statement, changing this value would permit us to explore the effect of this policy decision. This limit is not really a hard constraint on operations, because the trunk line can accommodate vastly more than ten waiting calls. Two variables are needed in the model logic. These are the number of calls currently waiting in each of the queues (NQ_Car-Stereo and NQ_Other). The sum of these variables is compared to Max_On_Hold in making the decision whether or not to issue a busy signal. Finally, four variables that are needed to collect information about system performance. These include the total number of calls received (Total Arrivals) and the number of these calls that receive a busy signal (Number_Busy) and, hence, are rejected at the switch. The ratio of the number of rejected calls to the number of calls received gives us the fraction of calls rejected. We also will count the number of calls of each type that exceed the limit of 1 min in queue (Car-Stereo_Too_Long and Other_Too_Long). If either of these (or both) are greater than zero, the we have not met the manager's requirement.

### 4.6 Random Number Generator

Every simulation package has a random number generator. The random number generator (technically called a *pseudo-random number generator*) is a software routine that generates a random number between 0 and 1. This number is then used in sampling random distributions. For example, let us assume that you have determined that a given process delay is uniformly distributed between 10 minutes and 20 minutes. Then every time an entity goes through this process, the random number generator would a generate a number between 0 and 1 and evaluate the uniform distribution formula that has a minimum of 10 and a maximum of 20 time units. As an example, let us assume that the generated random number is 0.7312. The delay time then the delay time would be 10+(0.7312)*(20-10) = 17.312 units.. So the entity would delay for 17.312 units in the simulation. Everything that is random in the simulation uses the random number generator as an input to determine values.

In our example model, we will use a physical random number generator—the roll of two dice. The probability distribution for the roll of two dice is seen in Figure 2. In our model, the roll will be the basis for every random delay and randomly assigned value in the model. We have four randomly and one assigned delays values in the model:

1. Time between arrivals of cars at the call center = (DICE * 0.333) mins.
2. The delay at the IVR unit = (DICE * 0.3) mins.
3. The delay for car-stereo call processing = (DICE * 2) mins.
4. The delay for other-product call processing= (DICE) mins.

5. The product type requested by the caller call (assuming on average of 16.7 % all calls are for car-stereo products)

$$= \begin{cases} \text{Car Stereo}, & \text{if DICE} \le 4 \\ \text{Other}, & \text{otherwise} \end{cases}$$

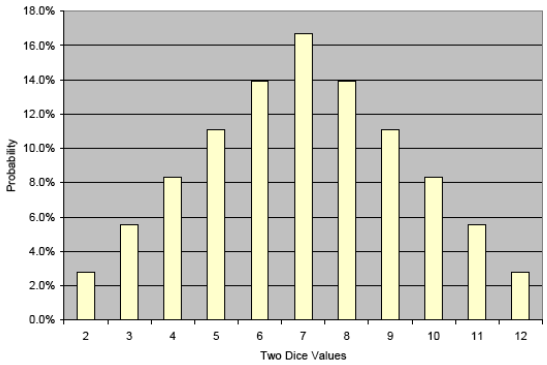


Figure 2: Distribution of Rolling Two Dice

### 4.7 The Clock and Calendar

The *clock* is a global variable (Current _Time) that carries the value of the current time in the simulation. The *calendar* is a list of events that are scheduled to occur in the future, i.e., at clock times later than the current clock time. In every simulation, there is only one calendar and it is ordered by the earliest scheduled-time first. In Section 5, it will become clear how the calendar works and why it is important in the simulation. At this point, just remember that, at any given point in time, every event that has already been scheduled to occur in the future is held on the calendar.

### 4.8 Statistics Collectors

Statistics collectors are a part of the simulation that collects statistics on conditions (such the number of units of the capacity of a resource in use), or the value of global variables, or certain performance statistics based on attributes of the entity. Three different types of statistics can be collected—*counts, time-persistent,* and *tallies.* Counts are very straightforward, they *count*. In our model, we defined four counts as global variables and we will collect statistics on these. *Time-persistent* statistical collectors give the time-weighted values of different variables in the simulation. A common variable to track is the utilization of a resource. In our model, we collect four different time-persistent statistics—the number of busy resources of the two resources that we have in the model and the number of entities in each of the corresponding queues. These are not strictly needed to assess the manger's criteria, but provide additional insight regarding the performance of the call center which might be useful if the manger's requirements are not met. *Tally statistics* are collected one observation at a time without regard to the amount of time between observations. In our model, we collect a very common statistic, which is the amount of time that an entity stays in the system. Since we assign the value of Start_Time to the attribute of the entity when it enters the call center, the value (Current_Time – Start_Time) is the total amount of time in the system. Again, this statistic provides additional insight on performance.

## 5 A WALK THROUGH THESE CONCEPTS

What we are going to do now is walk through a couple of steps in our simulation model. As you will see, we will track the state of the system, the entities on the calendar, the values of attributes and state variables, and the statistics we are collecting.

### 5.1 The State of The System at Noon

We assume initially that there are one car-stereo agent and two agents for other products. Figure 3 shows the image of the system at noon. The calendar of this system is made up of the entities that are scheduled to complete an activity with a specific time duration, as shown in Table 1. The two queues are shown with the entities in rank order in Tables 2 and 3 (we assume the calls are processed FIFO, i.e., in the order of arrival to the queue). Note that the event times are undefined. Each event record on the three lists also includes the values of the entities attributes when these are defined.

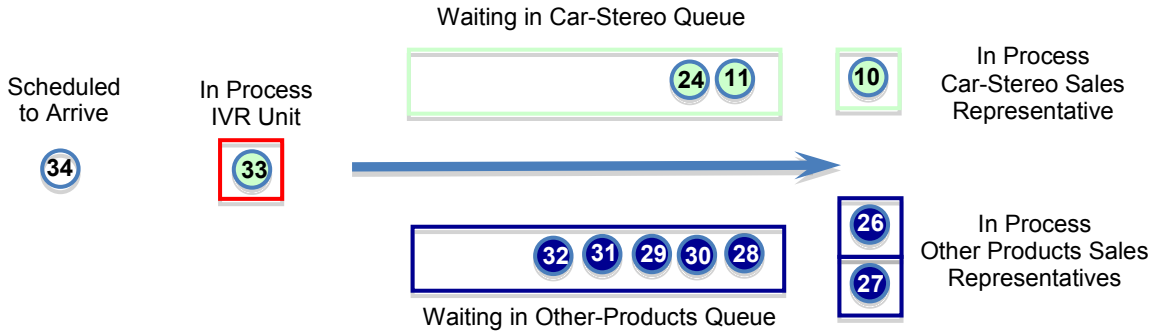


Figure 3: State of the System at Noon

Other important information concerns the clock, Current_Time, which is set to 12:00:00 PM. The statistics that we are tracking in the simulation have the values listed in Table 4 as of noon. The "Time/Obs" column gives the amount of time that we have been collecting the statistic (which is since 11:00 AM) for time-persistent statistics or the number of observations for tally statistics.

Table 1: The Calendar at Noon

| Entity | Event | Event Time | Product_Type | Start_Time | Begin_Wait |
|---|---|---|---|---|---|
| **34** | Arrive at Call Center | 12:01:40 PM | --- | --- | --- |
| **33** | Complete Service IVR | 12:02:40 PM | car-stereo | 11:59:00 AM | --- |
| **10** | Complete Service Car-Stereo | 12:04:00 PM | car stereo | 11:04:20 AM | 11:07:20 AM |
| **26** | Complete Service Other | 12:06:00 PM | other | 11:43:20 AM | 11:45:04 AM |
| **27** | Complete Service Other | 12:07:00 PM | other | 11:45:20 AM | 11:46:16 AM |
| **1** | End replication | 02:00:00 PM | --- | --- | --- |

Table 2: Car-Stereo Queue at Noon

| Entity | Event | Event Time | Product_Type | Start_Time | Begin_Wait |
|---|---|---|---|---|---|
| **11** | Begin Service | --- | car-stereo | 11:17:30 AM | 11:16:00 AM |
| **24** | Begin Service | --- | car-stereo | 11:37:20 AM | 11:39:07 AM |

Table 3: Other-Product Queue at Noon

| Entity | Event | Event Time | Product_Type | Start_Time | Begin_Wait |
|---|---|---|---|---|---|
| **28** | Begin Service | --- | other | 11:47:40 AM | 11:50:59 AM |
| **30** | Begin Service | --- | other | 11:51:40 AM | 11:53: 12 AM |
| **29** | Begin Service | --- | other | 11:50:00 AM | 11:53:47 AM |
| **31** | Begin Service | --- | other | 11:53:40 AM | 11:55:28 AM |
| **32** | Begin Service | --- | other | 11:57: 46 AM | 11:59:46 AM |

Table 4: Statistics at Noon

| Statistic | Value | Time/Obs |
|---|---|---|
| Busy Signal Count | 0 | --- |
| Completions (Car-Stereo) Count | 3 | --- |
| Completions (Other-Product) Count | 18 | --- |
| Excessive Wait (Car-Stereo) Count | 3 | --- |
| Excessive Wait (Other-Product) Count | 8 | --- |
| Representative Utilization (Car-Stereo) | 1.00 | 1:00:00 |
| Representative Utilization (Other-Product) | 1.00 | 1:00:00 |
| Average Number Waiting (Car-Stereo) | 2.447 | 1:00:00 |
| Average Number Waiting (Other-Product) | 2.847 | 1:00:00 |
| Average Waiting Time (Both Types) | 7.711 | 1:00:00 |
| Average System Time | 17.35 | 1:00:00 |

### 5.2 The State of The System at 12:01:40 PM

Here is where we get one of the key ideas about a discrete-event simulation. A discrete-event simulation progresses by advancing the clock to the time of the next event, instead of by uniform time-intervals. In this sense the simulation is said to be *event driven*. If in our example we had distinct time intervals of 1 second, we would go through 140 time intervals before anything would happen. Instead we go straight to the next scheduled event, which is the first event on the calendar, which is scheduled to occur at 12:01:40 PM. That event is the arrival of entity 34 at the call center.

This is how the event plays out. The clock is advanced to 12:01:40, entity 34 becomes the *active entity*, and the record for entity 34 is removed from the calendar. When this entity arrives at the call center, the first activity in the simulation is to set its attributes. Obviously, Start_Time is set to 12:01:40 PM. Product_Type is set by rolling the dice and using the result of the roll in the appropriate formula given in Section 4.6. We roll a 10, which is greater than 4, so the attribute value is set to "Other".

After the attributes are assigned, entity 34 encounters the switch. Since there are currently a total of seven calls waiting (two in the Car-Stereo queue and five in the Other-Product queue), the entity is directed to the IVR unit. We determine the duration of the delay at the IVR unit by rolling the dice and using the result the roll in the appropriate formula given in Section 4.6. We roll a 5, so the delay is 0.3(5)=1.5 min (1 min and 0.5*60=30 sec). So entity 34 is delayed until 12:01:40+00:01:30=12:03:10. Its record is returned to the calendar (after entity 33 and before entity 10) with this event time and the updated attributes. Neither of the queues is altered by this event.

To complete this arrival event, we need finally to create the entity for the next arrival event and put its record on the calendar. This will be entity 35. We roll a 5, so the interarrival time is 5/3=1.333 min (1.333*60=20 sec) and the record for this entity is inserted into the calendar with event time 12:01:40+00:00:20=12:03:00 (after entity 33 and before entity 34). The updated calendar at the completion of the arrival event is shown in Table 5.

Table 5: The Calendar at 12:01:40

| Entity | Event | Event Time | Product_Type | Start_Time | Begin_Wait |
|---|---|---|---|---|---|
| **33** | Complete Service IVR | 12:02:40 PM | car-stereo | 11:59:00 AM | --- |
| **35** | Arrive at Call Center | 12:03:00 PM | --- | --- | --- |
| **34** | Complete Service IVR | 12:03:10 PM | other | 12:01:40 PM | --- |
| **10** | Complete Service Car-Stereo | 12:04:00 PM | car stereo | 11:04:20 AM | 11:07:20 AM |
| **26** | Complete Service Other | 12:06:00 PM | other | 11:43:20 AM | 11:45:04 AM |
| **27** | Complete Service Other | 12:07:00 PM | other | 11:45:20 AM | 11:46:16 AM |
| **1** | End replication | 02:00:00 PM | --- | --- | --- |

The statistics can be updated at this point as well. (However, most simulation packages only update statistics when the value that is being tracked changes). None of the counts have changed. However, the clock has advanced, so all of the time-dependent statistics have changed (modestly) as well. To give you an idea how this is accomplished, consider the average number waiting in queue for the car-stereo sales representative. At noon, the simulation had been running for one hour and the average value was 2.447. From noon to 12:01:40 PM, the number of cars waiting in line for the menu board has been 2. So the new time-weighted average is ((2.447* 1:00:00) + (2 * 0:01:40)) / 1:01:40 . If we convert this formula to seconds, it becomes ((2.447* 3600) + (2 * 100)) / 3700 = 2.435. All time-dependent statistics are calculated by recursion in this way.

### 5.3 The State of The System at 12:02:40 PM

Lets process one more event, just so we've got the idea. Looking at the calendar in Table 4, the next event record has entity 33 completing service at the IVR unit. The clock is advanced to 12:02:40, entity 34 becomes the *active entity*, and the record for entity 34 is removed from the calendar. Next, the Begin_Wait attribute for this entity is set to the current clock time. The Product_Type attribute is "Car-Stereo", so the entity is sent to receive service from the car-stereo sales representative. Finding the sales representative busy with another call and two calls already waiting, entity 34 joins the queue at rank 3. The statistics are updated to reflect the passage of time and that's it.

## 6 INTERPRETING OUTPUT STATISTICS

Let us assume that our model has run from 11:00 AM to 2:00 PM. At 2:00 PM, we stop the simulation and we have all of our statistics calculated. The "answer" for the simulation is in Table 6.

Table 6: Statistics at 2:00:00 PM

| **Statistic** | **Value** | **Time/Obs** |
|---|---|---|
| Busy Signal Count | 7 | --- |
| Completions (Car-Stereo) Count | 10 | --- |
| Completions (Other-Product) Count | 53 | --- |
| Excessive Wait (Car-Stereo) Count | 8 | --- |
| Excessive Wait (Other-Product) Count | 53 | --- |
| Representative Utilization (Car-Stereo) | 0.882 | 3:00:00 |
| Representative Utilization (Other-Product) | 1.000 | 3:00:00 |
| Average Number Waiting (Car-Stereo) | 1.231 | 3:00:00 |
| Average Number Waiting (Other-Product) | 6.654 | 3:00:00 |
| Average Waiting Time (Both Types) | 16.955 | 3:00:00 |
| Average System Time | 26.056 | 3:00:00 |

Its tempting to conclude immediately that the call center is grossly understaffed and does not nearly approximate the manager's requirements. Seven out of (7+10+53)=70 (or 10%) of arriving phone calls are rejected at the switch; 8 of 10 (or 80%) of car-stereo calls and 100% of other-product calls experience waits longer that 1 min; and the average waiting time is nearly 17 minutes—far too long for customer satisfaction. Indeed, I know that I would hang up long before this, unhappy, and would perhaps even try another retailer. We can see that the resources are stressed—the car-stereo representative is busy 88% of the time and neither of the other-product representatives draw a breathe while not on the phone. The situation on this day is so very bad that the conclusion may be warranted.

But caution is advised. Suppose that this is just an exceptionally bad day? Shouldn't we really look that the call center on more than when day before we jump to this conclusion? Indeed, we should, and this is always the case in discrete event simulation. The numbers given in Table 6 are a random answer to the performance of the system. This is because there are random inputs to the system that give us this answer. The answer generated by only one run is not really an answer at all.

To make the point, let us take the following 100 rolls of the dice. You would think that 100 rolls of the dice would tell us the average. But if you take the 100 rolls in Table 7, the average is only 6.72. Is that close enough? Would you be willing to bet that the next 100 rolls would come up with an average of 6.72. Probably not.

Table 7: 100 Rolls of the Dice

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 5 | 8 | 6 | 5 | 4 | 10 | 6 | 7 | 8 |
| 5 | 9 | 8 | 7 | 8 | 6 | 3 | 8 | 7 | 6 |
| 9 | 9 | 8 | 8 | 6 | 8 | 9 | 7 | 10 | 5 |
| 2 | 10 | 11 | 8 | 6 | 8 | 7 | 3 | 8 | 8 |
| 4 | 6 | 8 | 11 | 2 | 4 | 8 | 9 | 8 | 5 |
| 9 | 3 | 8 | 7 | 2 | 3 | 9 | 10 | 7 | 7 |
| 3 | 9 | 5 | 7 | 7 | 7 | 9 | 4 | 8 | 7 |
| 4 | 10 | 7 | 4 | 10 | 8 | 4 | 8 | 9 | 7 |
| 3 | 6 | 6 | 3 | 6 | 3 | 10 | 9 | 7 | 4 |
| 6 | 8 | 5 | 9 | 12 | 6 | 8 | 6 | 4 | 2 |
| **Average: 6.72** | | | | | | | | | |

So what is the answer to this problem? If we really want to estimate the average value of the roll of the dice if we roll the dice 100 times, we need to run the simulation more times. Each time that we run a simulation is called an *iteration* or *replication*. So, as an example, let us say that we have run our dice throwing simulation for 30 iterations and Table 8 has the average values for each of those iterations.

Table 8: 30 Iterations of Throwing Dice 100 Times

| | | | | | |
|---|---|---|---|---|---|
| 6.72 | 6.95 | 6.78 | 7.14 | 6.62 | 6.81 |
| 6.75 | 7.17 | 6.62 | 7.3 | 6.92 | 7.04 |
| 6.79 | 7.13 | 7.17 | 7.12 | 6.82 | 7.29 |
| 7.13 | 7.26 | 7.19 | 6.52 | 6.8 | 7.3 |
| 6.95 | 6.96 | 6.78 | 7.17 | 7.13 | 6.68 |
| **Average: 6.967** | **Standard Deviation: 0.22992** | | **95% Confidence Interval: [6.8847,7.0493]** | | |

If we are simply trying to estimate the average of 100 rolls of the dice, we do not need to worry about the standard deviation for each iteration. However, we do need to worry about the standard deviation of the averages from each iteration. As is shown in Table 8, the average of the averages (so to speak) is 6.967. The standard deviation of those 30 averages is 0.22992. With that information, we can calculate a *confidence interval*.

A *confidence interval* is a statistical measure we use to bound some statistic. The level of confidence (95% in our example) is the statistical probability that the statistic that we are considering lies in the interval. So, the interpretation of the 95% confidence interval of [6.8847,7.0493] would be, "If we repeatedly roll the dice 100 times, on average across repetitions 95 rolls will result in a mean which lies between 6.8847 and 7.0493." Most statistical and simulation packages automatically calculate confidence intervals for you. Even Microsoft Excel has a function to calculate confidence intervals.

So in our example, we want to run 30 iterations so that we can have good confidence intervals for each of our statistics. (Although we will not get into the topic in this paper, one should run 30 iterations (or more) if you can in order to get good confidence intervals.) Table 9 shows the confidence intervals for each of our statistics. Although these do not fundamentally alter our conclusions based on a single run, we know now with some certainty that this wasn't just a bad day.

Table 9: Confidence Intervals after 30 Iterations

| Statistic | Value | LowerCI | Upper CI |
|---|---|---|---|
| Busy Signal Count | 9.933 | 8.183 | 11.683 |
| Completions (Car-Stereo) Count | 11.1 | 10.43 | 11.77 |
| Completions (Other-Product) Count | 50.3 | 49.12 | 51.48 |
| Excessive Wait (Car-Stereo) Count | 9.467 | 8.267 | 10.667 |
| Excessive Wait (Other-Product) Count | 50.2 | 49.03 | 51.37 |
| Representative Utilization (Car-Stereo) | 0.897 | 0.847 | 0.947 |
| Representative Utilization (Other-Product) | 0.999 | 0.999 | 0.999 |
| Average Number Waiting (Car-Stereo) | 1.782 | 1.312 | 2.252 |
| Average Number Waiting (Other-Product) | 6.518 | 5.908 | 7.128 |
| Average Waiting Time (Both Types) | 17.57 | 16.85 | 18.29 |
| Average System Time | 26.367 | 25.667 | 27.057 |

## 7 FINDING WAYS TO IMPROVE A SYSTEM

What we have accomplished up to this point is the analysis of a system based on an initial design calling for one car-stereo and two other-product sales representatives. We have learned that this design falls far short the manager's performance requirements. Since there are excessive waits for both call types, we also have learned that we will need to add *at least* one new representative to each of the corresponding resource pools. So let us run some alternative scenarios to determine how many. Let's experiment with the model!

### 7.1 Scenario 1: Increasing the Number of Sales Representatives

We tried increasing each server pool by one sales representative and found that this reduced the number of rejected calls to zero, but still yielded excessive waiting times for each type of call. So we tried increasing each server pool by two representatives. This again improved performance, but still fell short of the manager's stringent requirements. Continuing to increase the number of representatives incrementally, the requirements were met with a minimum of 4 car-stereo representatives and 6 other-product representatives. The results of this simulation are shown Table 10.

The statistics show that this design is the Cadillac of call centers—all calls are received and there is essentially no waiting. Callers receive superb service and this certainly translates into increased sales and goodwill. But the price is steep in terms of the resources required—a total sales force of 10 representatives. The utilization of the representatives is low (there is almost always someone available to take your call immediately), with the car-stereo representative busy only about 16 min out of each hour and the other-product representatives busy only about 26 min out of each hour.

The decision on whether or not the improved performance is worth the increased cost is complex and well beyond the scope of this paper. We would need to consider the hourly wages of the sales representatives, retention rates and the expense associated with recruiting and training (the Cadillac center certainly represents a less stressful work environment), the expected revenue from accepting an additional 10 calls into the center in a three hour period (noting also that an average waiting time of over 17 min in the original case is unrealistic and many of these callers simply will hang up before receiving ser-

vice), the call-back rate both of folks receiving a busy signal and folks balking because of excessive waiting, and the value of goodwill lost or gained through superior service (and how this translates into product pricing, future sales, and market share).

Moreover, the idle time in the Cadillac system may or may not be used productively in performing other tasks. Indeed, when the center is not congested, the duration of calls is likely to increase as sales representatives have the time to engage in cross selling ("if you are looking for that particular car stereo, we have the perfect set of speakers you might also want to consider…"). With adequate knowledge, many of these considerations could be incorporated into a more complex simulation and improved economic analysis. Indeed, one of the great benefits of modeling and simulation is improved understanding of the system, what's important and what's not, and what questions (however complex) we should be asking.

Table 10: Confidence Intervals for Scenario 1

| **Statistic** | **Value** | **Lower CI** | **Upper CI** |
|---|---|---|---|
| Busy Signal Count | 0 | 0 | 0 |
| Completions (Car-Stereo) Count | 13.433 | 12.273 | 14.593 |
| Completions (Other-Product) Count | 65.667 | 64.777 | 66.857 |
| Excessive Wait (Car-Stereo) Count | 0.067 | 0 | 0.207 |
| Excessive Wait (Other-Product) Count | 0.067 | 0 | 0.157 |
| Representative Utilization (Car-Stereo) | 0.267 | 0.247 | 0.287 |
| Representative Utilization (Other-Product) | 0.437 | 0.427 | 0.447 |
| Average Number Waiting (Car-Stereo) | 0 | 0 | 0 |
| Average Number Waiting (Other-Product) | 0.001 | 0.001 | 0.001 |
| Average Waiting Time (Both Types) | 0 | 0 | 0 |
| Average System Time | 10.127 | 9.977 | 10.277 |

### 7.2 Scenario 2: Cross-training Sales Representatives

It is well known that having a single queue feeding multiple resources is more efficient that having each individual resource fed exclusively by its own dedicated queue. Think of the difference between waiting at an airline check-in counter, where everyone (flying economy class) waits in a single line for the next available agent, versus waiting in line to checkout at a grocery store, where you need to make the (often maddening) choice about which line you should join. The reason is obvious when you think about it—at the airline counter there is never an idle agent while there is anyone in line; at the grocery store, there are times when one check-out register is available, but you are in the wrong line waiting.

In our example, we have two lines, one for each type of call. The sales representatives are specialists. If instead we were to "cross train" all representatives to process both call types, all calls could wait in a single queue for the next available representative. In our simulation, we implemented this scenario by simply making all arriving calls other-product type, while retaining the same volume of inbound calls. After some experimentation we found that we could cut the total workforce in half (from 10 specialist to 5 generalist representatives) and still achieve the manager's requirements. The results of this simulation are shown Table 11.

Table 11: Confidence Intervals for Scenario 2

| **Statistic** | **Value** | **LowerCI** | **Upper CI** |
|---|---|---|---|
| Busy Signal Count | 0 | 0 | 0 |
| Completions (Car-Stereo) Count | --- | --- | --- |
| Completions (Other-Product) Count | 78.833 | 77.373 | 80.293 |
| Excessive Wait (Car-Stereo) Count | --- | --- | --- |
| Excessive Wait (Other-Product) Count | 1.600 | 0.92 | 2.28 |
| Representative Utilization (Car-Stereo) | --- | --- | --- |
| Representative Utilization (Other-Product) | 0.625 | 0.605 | 0.645 |
| Average Number Waiting (Car-Stereo) | --- | --- | --- |
| Average Number Waiting (Other-Product) | 0.053 | 0.033 | 0.073 |
| Average Waiting Time (Both Types) | 0.053 | 0.033 | 0.073 |
| Average System Time | 9.058 | 8.949 | 9.168 |

Performance is comparable to the specialist case, with no calls rejected at the switch and something like 2% of all calls experiencing waiting times greater than 1 min (an estimated 1 to 2 calls of the estimated total of 79 to 80 calls arriving). The utilization of the generalist pool is about 62-65%, which should allow time for cross-selling without a great deal of stress.

At first, it may seem like we are getting something for nothing—the total number of representatives required is actually less than the number needed to serve just the other-product calls in the first scenario! A little reflection explains why, however. The largely idle car-stereo representatives are now taking other product calls when needed, and vice versa. Moreover, cross-training is not free and may not afford the uniformly knowledgeable service available through specialization. As before, the decision on whether or not to cross train some or all representatives is well beyond the scope of this paper. But the dramatic reduction in workforce suggested by the simulation makes this an option well worth investigating.

### 7.3 Generating Other Scenarios

This simulation and any other simulation can be used to evaluate many different scenarios if the person creating the model allows some flexibility in the model structure. One also has to consider the amount of time it takes to run a new scenario. In a large scale simulation, to run and evaluate a new scenario could take several days.

In our example, there are other scenarios we could play out in the simulation. For example, we noticed early on that the manager's requirement bounding the number of calls rejected at the switch is satisfied with a sales force of just two car-stereo and three other-product representatives. By forcing rejection at the switch at a lower degree of congestion, we might be able to meet all requirements with a smaller staff. We'll leave exploration of this and other options for a rainy day.

## 8 WHAT HAVE WE LEARNED? EPILOGUE

Now that we have run through our example, you should understand the mechanics of how a discrete-event simulation works. You should also sense that many different types of activities are required to perform a simulation study and that each of these activities must be performed competently in order to wrench understanding from what is otherwise just information and data. The scope of these activities are suggested broadly in Figure 4.

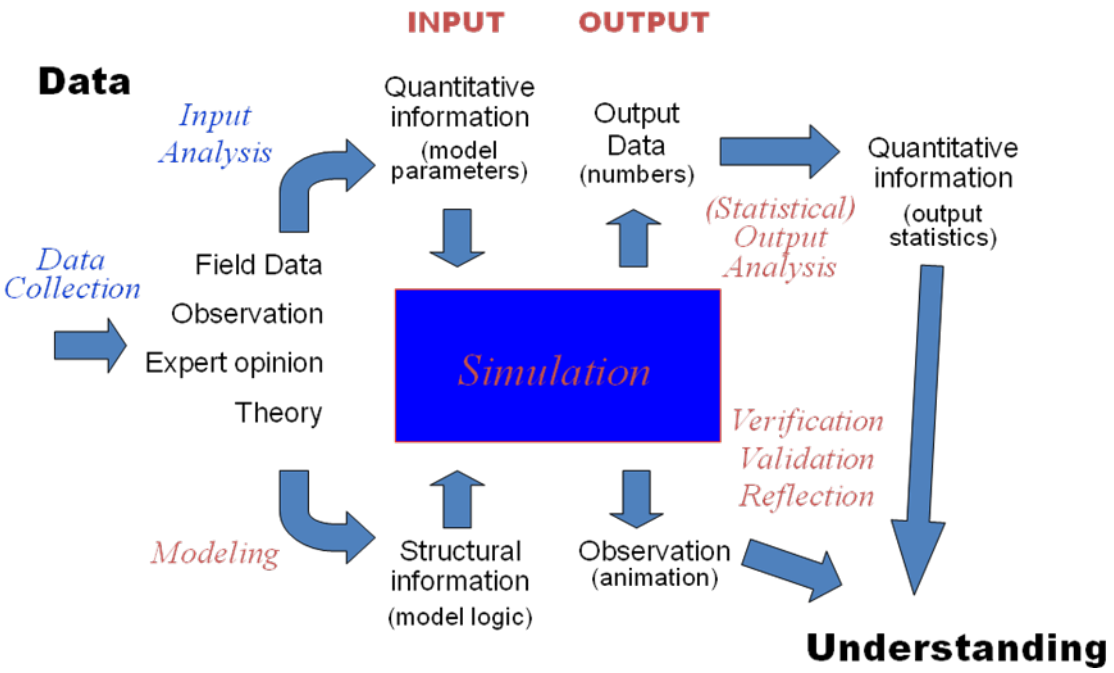


Figure 4: Activities in a Simulation Study

What else have we learned? This exercise points out several things about simulation in general. First, simulation can mimic the *dynamic* behavior of a system. That is what it is built to do. Regardless of how complex a system may be, it is

likely that a simulation *expert* will be able to create a model that will evaluate it. However, the more complex a system is, the longer it takes to model, run, and evaluate. But do not be discouraged, there are very good simulation people available to model large systems. Second, you (or the person analyzing the system) must have a good understanding of simulation statistics. It is important during the creation of the model so that input distributions are used properly. It is important during the analysis of the output statistics so that the output is not misinterpreted. Mistakes regarding either the inputs or the outputs will invalidate the simulation. Third, to analyze a system, simulation is used to evaluate different scenarios. It does not choose the best scenario for you. This may seem to be a problem, but most managers have no shortage of scenarios to evaluate. The trade off for this is that you can analyze the *dynamics* of the system and not just the average behavior. Fourth, the scenarios that you do choose are generated by you and not the system. This is where familiarity with the system under study and a familiarity with system dynamics concepts are very valuable. This is, of course, simply an introduction. Through this conference and interaction with simulation professionals, you can get a deeper understanding of simulation and what it can do for you.

## REFERENCES


Banks, J., J.S. Carson II, B.L. Nelson, and D.M. Nicol. 2000. *Discrete Event System Simulation*, 3rd ed., Prentice-Hall.

Chung, J-S., and K. P.White, Jr. 2008. "Cross-Trained vs. Specialized Agents in an Inbound Call Center: A Simulation-Based Methodology for Trade-Off Analysis," *Journal of Simulation*, **2**(3)162-169.

Law, A.M. 2007. *Simulation Modeling and Analysis*, 4th ed., McGraw-Hill.

Kelton, W.D., R. Sadowski, and N.B. Swets. 2010, *Simulation with Arena,* 5th ed., McGraw-Hill.

Ingalls, R.G, and C. Kasales. 1999. CSCAT: Compaq Supply Chain Analysis Tool. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Ingalls, R.G. 1998. The Value of Simulation in Modeling Supply Chains. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Ingalls, R.G. 2008. Introduction to Simulation. In *Proceedings of the 2008 Winter Simulation Conference*, ed. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Ingalls, R.G., and C. Eckersley. 1992. Simulation Issues in Electronics Manufacturing. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J.J. Swain, D. Goldsman, R.C. Crain, and J.R. Wilson. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Shannon, R.E., 1975. *Systems Simulation – The Art and Science*, Prentice-Hall.

White, K. P., Jr. 2007. "Modeling and simulation", *Encyclopedia of Electrical Engineering and Electronics*, Wiley, New York, 13:404-417.


## AUTHOR BIOGRAPHIES


**K. PRESTON WHITE, JR.,** is Professor of Systems Engineering at the University of Virginia. He received the B.S.E. (1970), M.S. (1972), and Ph.D. (1976) degrees from Duke University. He has held faculty appointments at Polytechnic University and Carnegie-Mellon University and served as Distinguished Visiting Professor at Newport News Shipbuilding and at SEMATECH. He is a member of INFORMS and INCOSE and a senior member of IEEE and IIE. He sits on the Advisory Board of VMASC and is a member of the NASA Engineering Statistics Team. His research interests include stochastic simulation and applications to requirements verification. He is a member of the WSC Board of Directors. His mail address is <kpwhite@virginia.edu>.

**RICKI G. INGALLS** is Associate Professor and Site Director of the Center for Engineering Logistics and Distribution (CELDi) in the School of Industrial Engineering and Management at Oklahoma State University. He is also Vice President of Diamond Head Associates, Inc. He joined OSU in the Fall of 2000 after 16 years in industry with Compaq, SEMATECH, General Electric and Motorola. He has a B.S. in Mathematics from East Texas Baptist College (1982), a M.S. in Industrial Engineering from Texas A&M University (1984),and a Ph.D. in Management Science from the University of Texas at Austin (1999). His research interests include the supply chain design issues and the development and application of qualitative discrete- event simulation. He is a member of IIE and Program Chair for WSC09. His email address is <ricki.ingalls@okstate.edu>.