4

System Definition

"How do you define that?"

4.1 Introduction

Once the problem has been properly formulated and the project plan is complete, the simulation practitioner should direct his or her attention toward defining the system that should be modeled. For our purposes, a system is a group of components that receives inputs and delivers outputs. The components determine how the system converts inputs to outputs. The system may exist in an environment that could also potentially affect the inputs and outputs. The system definition and model formulation process consists of determining:

- The system classification
- How much of the system to model
- What components and events to model
- What input data to collect
- What output data to generate with the model

Note that we are going to define our system and formulate the model at the same time. In many other simulation texts, these two processes are treated completely independently. By defining our system and formulating the model nearly simultaneously, we are in a better position to understand how the system components will be modeled. In this chapter we focus on what we are going to model. In the following model translation chapter, we discuss the actual process of how we are going to model these components.

4.2 System Classifications

One of the first steps that the practitioner must perform in the system definition phase is to classify the system. Systems can be classified with respect to two different dimensions. First, a system may be discrete, continuous, or combined. Second, a system is either terminating or nonterminating. These classifications are a significant issue; they affect how the practitioner will conduct the modeling and analysis for the project (Law and Kelton, 2000).

4.2.1 Discrete versus Continuous versus Combined

The classification of the system as being discrete, continuous, or combined is a function of how the simulation clock will function. As you may recall from Chapter 1, the simulation clock manages system events. System events are occurrences that change the status of some component in the system.

4.2.1.1 Discrete Event Systems

In Chapter 1, "Introduction," the example that was used was a discrete system. The example system events occurred according to discrete jumps in the time clock. In that example, the system events were arrivals, service starts, and service ends. These events occurred at particular points in time on the event list. Between events on the event list, the system did not change with respect to the number of entities in the system. So, systems that jump between events are considered as discrete event systems. It is important to note that the type of entities in the system can cause the way that the system jumps between events. Entities that are also individual or discrete in nature promote the advancement of the clock in discrete jumps. Generally speaking, any system that uses people as an entity will be a discrete event system.

Examples of discrete event systems include:

- Stores
- Service centers
- Manufacturing facilities
- Transportation centers

4.2.1.2 Continuous Event Systems

In contrast to discrete event systems, in continuous event systems, some event is always occurring. This means that the status of some component in the system is continuously changing with respect to time. Systems that are continuous usually involve some sort of fluid or fluid-like substance. Fluid-like substances could include any type of small-particle, high-volume material that flows like a fluid. These types of materials are usually measured by weight rather than count. An example of a fluid-like substance is coffee. Coffee starts out as a bean and then is continuously processed, eventually resulting in a grain.

Continuous event systems must be modeled with differential equations. Because of the additional complexity presented by the use of differential equations, these models are typically more difficult to model accurately. The fluid or fluid-like materials are not modeled as entities but as either volume or weight. The volume or weight flows through the model.

Examples of continuous event systems include:

- Water treatment plants
- Chemical industries not including distribution points

4.2.1.3 Combined Event Models

Combined event models contain both discrete and continuous components. This means that the entities in the model may exist in an individual countable form in one part of the model and a fluid-like form in another part of the model. This type of situation occurs in many processing plants where the fluid or fluid-like substance is canned or packaged. Examples can be found in:

- Food industries
- Chemical distribution points

A specific example of a combined model in the food industry is the coffee plant that was previously discussed. The system begins with fluid-like beans in storage tanks. The beans go through a series of roasting and grinding processes. The fluid-like coffee particles are eventually packaged into either cans or bags. As soon as the coffee is canned or packaged, that part of the model becomes discrete. The individual can then flows though the system as an entity until it departs.

In the chemical industry, a petroleum product simulation can begin with oil in a tank. Obviously, the oil is a fluid. It must therefore be modeled using a continuous event model. The oil will eventually be transported by tanker trucks. The process of filling and emptying changes the continuous model into a discrete model. In this case, the discrete component is the tanker truck. When the tanker truck is filled, it can individually be modeled for further processing.

Combined event systems are typically the most difficult type of system to model. The practitioner has to handle not only the increased complexity of the continuous event modeling but also the transformation of the fluid or fluid-like material to discrete entities. The practitioner must also decide how to handle the output measures of performance. Will they be based on the continuous portion of the model, the discrete portion of the model, or the interface between the continuous and discrete portions of the model?

4.2.2 Terminating versus Nonterminating

The second means of classifying the system is whether it is terminating or nonterminating in nature. Terminating and nonterminating systems are distinguished by:

- Initial starting conditions
- Existence of a natural terminating event

4.2.2.1 Initial Starting Conditions

The status of the system at the beginning of the period of interest is one means of distinguishing between a terminating and a nonterminating system. Terminating systems generally start each time period without any influence from the previous time period. This means that the system cleans itself of entities between time periods. Many systems that utilize customer-type entities are considered terminating-type systems. These systems do not have any customers left in the system from the previous time period. A bank, for example, does not let customers stay in the bank overnight. This means that each new day or time period, the system starts empty.

In contrast to the terminating system, the nonterminating system may begin with entities already in the system from the previous time period. In this case, the system may close with entities still in the system. At the beginning of the next period, the system starts with whatever was left from the previous period. Even though the system actually closes between time periods, it starts again as though it had never closed. Another type of nonterminating system actually has no beginning time period and no closing time. These types of systems just continuously run. In this case the system never stops, so it may never clean itself of entities.

4.2.2.2 Existence of a Natural Terminating Event

A second means of identifying whether a system is terminating or nonterminating is the existence of a natural terminating event. This may be the shutting down of the system at a particular point in time or the end of a busy period that is of specific interest to the practitioner. Examples of systems shutting down are service systems that close at the end of the day. Examples of systems that have busy periods are those that exhibit a high degree of activity during specific meal times. The existence of naturally occurring terminating events means that these systems may be classified as terminating systems, subject to other terminating system requirements.

4.2.2.3 Types of Terminating Systems

There are many different types of terminating systems. However, in general, terminating systems must:

- Have a natural terminating event
- Not keep entities in the system from one time period to the next

Examples of terminating-type systems include:

- Stores
- Restaurants
- Banks
- Airline ticket counters

Stores and restaurants generally close at the end of the day. This constitutes a terminating event. All of the customers are forced out of the store. So, each new time period, the system starts empty.

Banks are similar to stores; however, the period of interest would be shorter, perhaps the period between 10 a.m. and 2 p.m. The terminating event may be considered as the end of the busy period at 2 p.m. In many cases, this would be the end of the lunch period, at which time the customers would have to return to work.

Airline ticket counters can also be modeled as terminating systems. Here, the period of interest might be the time between one batch of planes arriving and the next batch of planes departing. Thus, the terminating event could be the departure of the batch of planes. Several time period cycles may exist in a single day as a result of the use of hub-type air transportation systems.

4.2.2.4 Types of Nonterminating Systems

There are also many different types of nonterminating systems. In general, nonterminating systems can either:

- Have a terminating event but keep entities in the system between time periods
- Not have a terminating event and run continuously

This means that a number of systems that may actually appear to be terminating systems are actually nonterminating systems because they keep entities in the system between time periods. It is also possible for a nonterminating system to appear as a terminating system because the system may be temporarily emptied of entities during nonbusy periods. Examples of nonterminating-type systems include

- Most manufacturing facilities
- Repair facilities
- Hospitals

Many manufacturing systems run continuously 24 h/day, 7 days/week. The only time they shut down is for annual maintenance. Other manufacturing systems do shut down after either one or two shifts per day. However, these systems probably keep work in progress between shutdowns.

Repair facilities that keep the end item in the facility between shutdown periods are nonterminating systems. Even though they terminate at the end of each day, the work not repaired each day remains in the system. An example of this type of system is a car dealership service center.

Hospitals do not normally close. Although the activity may be reduced during the early hours of the morning, because the system does not close, it can be considered as a nonterminating-type simulation.

4.2.2.5 Statistical Analysis Approach Based on Type of Simulation

As we point out in Chapter 5, "Input Data Collection and Analysis," there is a great difference in the manner in which terminating and nonterminating systems are analyzed. Terminating system analysis is significantly easier to perform than nonterminating analysis. For this reason, many practitioners incorrectly model and analyze nonterminating systems as terminating systems. Practitioners who are not

confident with the nonterminating system analysis approach may attempt to modify the system in order to use the less demanding terminating system approach. The usual technique is to look at only a small period during the long nonterminating system run and to use a terminating system analysis approach.

4.3 High-Level Flow Chart Basics

An essential tool for defining the system is a high-level flow chart. A high-level flow chart helps the practitioner obtain a fundamental understanding of the system logic. It graphically depicts how the major components and events interact. It also illustrates how the input and output data play a role in the model. To help insure that your flow chart is properly developed, we begin this section with basic information on the design of high-level flow charts.

The components and logic of the system should have been previously observed in the orientation activities associated with the problem statement phase. Because the simulation flow chart is a high-level flow chart, the practitioner should endeavor not to include so much data that it is difficult to follow. The chart is intended only to provide general information as to how the process flows. It is understood that the actual programming of the simulation model will require additional detail.

Unfortunately, the development of even relatively simple high-level flow charts requires a certain level of discipline. All too often, practitioners attempt to sit down and begin modeling the system without a functional high-level flow chart. This approach usually has an unhappy ending. Without the flow chart it is doubtful that the practitioner possesses a fundamental understanding of the model. The writing of the simulation program is difficult enough. This process will become significantly more difficult if the practitioner attempts simultaneously to address the logic and programming issues.

4.3.1 Standard Flow Chart Symbols

There are four basic flow chart process symbols. These are the:

1. Oval
2. Rectangle
3. Tilted parallelogram
4. Diamond

There are specific ANSI standards for the use of these flow chart symbols. For practical use by simulation practitioners, these standards can be reduced to a couple of common-sense guidelines. The first guideline is that the symbols should be arranged so that the sequence of processes flows downward and to the right as much as possible. The second guideline is that any given symbol should have only one connecting path into the symbol and only one connecting path out of the symbol. The only exception to this rule is the decision icon, which has two different output paths. However, even with the decision icon, only one output path can be taken at a given time. Last, it is helpful to try to keep the flow chart on as few sheets of paper as possible. If additional detail is required, it should involve a lower-level flow chart of the particular process, not additional detail in the high-level flow chart.

4.3.1.1 Start and Stop Oval

The oval symbol is used to designate both the start and stop processes. The start symbol is the first symbol that should be present on the flow chart. The start symbol generally has only a single output connector. The stop or end symbol is normally the last symbol present on the flow chart. Even if the process has several different possible paths, there should still only be one stop or end symbol. All of the processes should terminate in this one symbol. The start oval is illustrated in Figure 4.1. The stop oval is identical except for labeling.

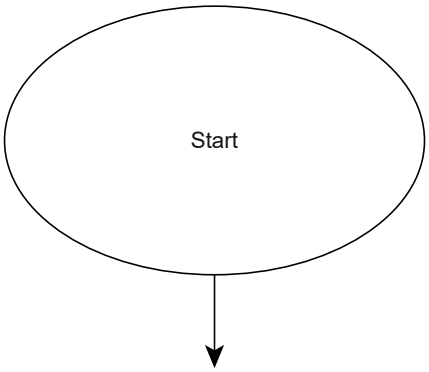

FIGURE 4.1 Start flow chart symbol.

4.3.1.2 Process Rectangle

The rectangle is used to represent general-purpose processes that are not specifically covered by any of the other flow chart symbols. The process rectangle is normally entered from the top or the left side. It is exited from either the bottom or the right side. In a high-level flow chart, a service time delay would be a common example of a process (Figure 4.2).

4.3.1.3 Input/Output Tilted Parallelogram

The tilted parallelogram is used for processes that involve some sort of input or output. The tilted parallelogram is normally entered from the top or the left side (Figure 4.3). It is exited from either the bottom or right side. An example of an input process symbol in a simulation program flow chart would be the creation or arrival of entities into a system.

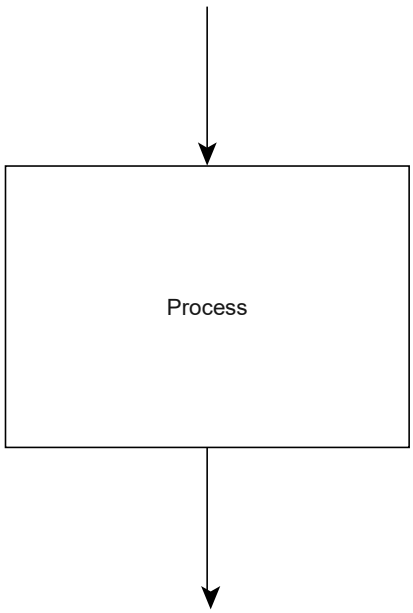

FIGURE 4.2 Process flow chart symbol.

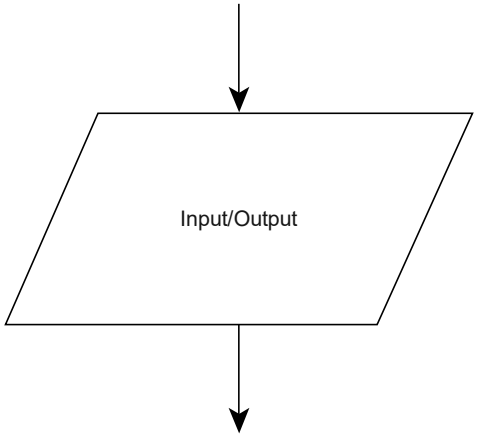

FIGURE 4.3 Input/output process flow chart symbol.

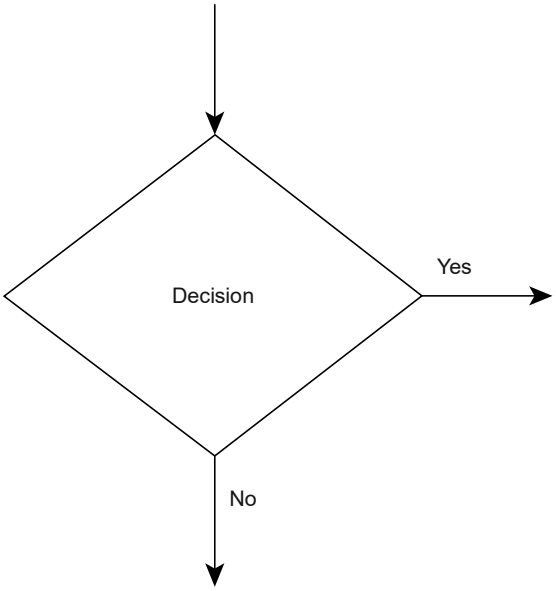

FIGURE 4.4 Decision flow chart symbol.

4.3.1.4 Decision Diamond

The diamond is used to represent a decision in the flow chart logic. The diamond has one input connector but two output connectors (Figure 4.4). The single input connector should come into the top vertex of the diamond symbol. The output connectors can leave through either of the side vertices or the bottom vertex. The output connectors must specifically be labeled as either true or false or yes or no. Because these responses are mutually exclusive, only one output path may be taken at a given time.

4.3.2 Sample Flow Chart

The sample flow chart in Figure 4.5 represents a simple system example such as the one used in Chapter 1. This is a single-queue, single-server system.

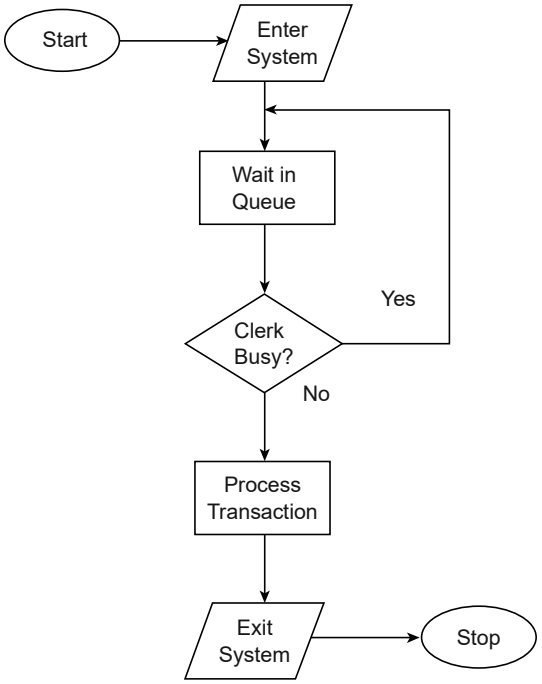

FIGURE 4.5 Sample flow chart.

The system flow chart begins with the start oval. The next symbol is an input/output tilted parallelogram, which represents the customer entering the system. After the customers enter the system, they enter into the queue. Waiting in the queue is a process represented by a rectangle. The customer next determines whether or not the clerk is busy. This operation is represented by a diamond. If the clerk is not busy, then the customer undergoes the transaction process represented by another rectangle. However, if the clerk is busy, the customer stays in the queue. When the customer finally finishes the transaction, the customer exits the system. This operation is represented by another tilted parallelogram. The flow chart finishes with an end oval symbol.

Note how the flow of the process is to the right and down as much as possible. The only place where the flow is not down is when the customer remains in the queue. Also note how most of the symbols have only one entry and one exit. The only exceptions are the start with a single exit, the stop with a single entrance, and the decision. Even though the decision has two exits, only one exit can be taken at a time. Either the clerk is busy or the clerk is not busy. Note also how when the clerk is busy that the flow line does not go directly back to the queue. Instead, the flow line intersects with the flow line that goes into the queue. This preserves the principle that the queue symbol can be entered in only one place.

In this particular example, the flow chart utilized six symbols. The flow chart for a more complicated system could easily require tens if not hundreds of flow chart symbols. As long as the practitioner strives to keep the flow chart organized, a large number of symbols can be managed. Keeping the flow chart organized means flowing to the right and down and only entering and exiting each symbol in one place at a given time.

4.4 Components and Events to Model

Now that we have reviewed the basic concepts of creating a flow chart, we can turn our attention toward what components and events should be modeled and represented in our high-level flow chart.

In a small independently operating system, there is little question that the practitioner should attempt to model the entire system. However, in a large interrelated system, it may be very difficult to identify which parts of the system to model. On one hand, the model must contain enough content so that the behavior of the system is properly understood. On the other hand, only a limited amount of time is available for most projects. Thus, in the beginning it is natural that some question may exist as to the appropriateness of the initial model. The suggested approach is to begin with a coarse model and add refinements as you proceed. It is better to have a higher-level model that can be used for more limited analysis than it is to have a lower-level model that cannot be used for analysis. Many practitioners have also found that the increased modeling demands of the lower-level model do not allow much time for analysis or reporting activities.

4.4.1 Components

Service models and manufacturing system models will naturally contain different system components. Some more common system components that may be modeled include (Kelton et al., 2002):

- Personnel
- Machines
- Transporters
- Conveyors

4.4.1.1 Personnel

In service-related systems, personnel may include:

- Sales clerks
- Customer service representatives

In manufacturing systems, personnel would probably include:

- Machine operators
- Material handlers

4.4.1.2 Machines

Machines in service-related systems may include:

- Computer or network systems
- Automatic teller
- Ticket machines
- Scanners
- X-ray machines

Machines in manufacturing-related systems could include:

- Computer numerically controlled mills
- Machining centers
- Lathes
- Turning centers
- Robots

4.4.1.3 Transporters

A transporter is any kind of vehicle that is used to move entities from one location to another. In transportation service models, transporters may include vehicles such as

- Airplanes
- Buses
- Trains

In manufacturing-related systems, there may be

- Forklifts
- Hand trucks
- Dollies
- Automatically guided vehicles

Note that there is also a distinction between free-path and fixed-path transporters. Free-path transporters can go between different locations without respect to following a specific path or the presence of other vehicles. A typical example of a free path transporter is a fork lift. Fork lifts typically can move between points without a specific track. Forklifts can also go around other transporters because they are not restricted to movement on rails. On the other hand, fixed-path or guided transporters must follow a specific rail or track and can be affected by the presence of other vehicles. A typical example of a fixed-path transporter would be a rail train. The forward movement of any rail train is dependent on clear track ahead of the rail train.

4.4.1.4 Conveyors

Conveyors are devices that can move entities between locations without the conveyor device itself physically moving. The conveyor actually physically stays in the same place, but the belt or rack does the moving. Examples of conveyors in service systems include:

- Moving sidewalks
- Escalators
- Chair lifts

Conveyors in manufacturing systems include:

- Overhead crane systems
- Fixed production assembly lines

Conveyors can also be classified as:

- Nonaccumulating
- Accumulating

Nonaccumulating conveyors maintain the spacing on the conveyor belt or track between entities. New entities can be placed on the conveyor only if there is sufficient space between other entities. Nonaccumulating conveyors are commonly circular in shape. This means that if the entities cannot be removed, they can ride around the conveyor an unlimited number of times. One typical example of a nonaccumulating conveyor is the suspension system that is used to retrieve clothes in a dry cleaning store.

Accumulating conveyors, on the other hand, allow the entities to become compressed on the conveyor belt. This usually means that there is some mechanism on the conveyor that causes the relative position between the entities on the belt to be altered. Accumulating conveyors are also usually linear in shape and transport entities in only one direction. An example of an accumulating conveyor is the cash register point in many grocery stores. The clerk operating the belt can run the system until the goods are compressed at the end of the belt.

Some conveyors appear to have a combination of characteristics. The most immediate one that comes to mind is the baggage conveyor in the baggage claim area of a airport. Here, bags can come down a chute onto the main circular conveyor. The impact of the bags can cause the density of the bags in a section of conveyor belt to increase. On the other hand, the belt is circular, so there is no other specific mechanism that would cause the bags on the belt to become compressed.

4.4.2 Processes and Events

The practitioner must also decide what system events should be included in the model. One way of determining which processes to model is to include any process that is capable of being performed differently over time.

4.4.2.1 Service System Processes and Events

In a service system, customers may wait in line for different amounts of time, have different services or amounts of goods to purchase, and the customers may pay differently. Thus, the model must at least include:

- Arrival of customers at a processing area
- Customer queue behavior
- Service processing
- Payment for the goods

4.4.2.1.1 Arrival of Customers at a Processing Area

The event of customers initially arriving into the system is perhaps one of the most important events in service-type models. In simple models, some practitioners will not properly model the arrival process. As we will see in the input data collection section of this chapter, it is not only important to model the arrival of the arrival batches, but also the size of the arrival batches.

4.4.2.1.2 Customer Queue Behavior

Most service-type systems include waiting queues for customers. Customer queues can either be

- Parallel queues
- Single snake queues

Parallel queues are found in systems that have multiple server resources. Each server has a dedicated queue. Customers enter the tail end of one of several queues and proceed to the head of the queue as previously queued customers leave the queue to receive service. Once at the head of the parallel queue, the customer will wait until the dedicated server is available for processing.

In contrast to the multiple parallel queues, a single snake queue is often used to model complex systems. Snake queues earn their name for their back and forth layout. This layout is used in order to make the best use of available space. With a snake queue, a single queue holds customers for a number of server resources. Once the customer reaches the head of the queue, the customer will wait until the first server resource is available for processing. An issue with snake queues with multiple servers is what occurs when one or more resources are available. In this situation, the system can operate in different ways. These are:

- Preferred order
- Random

In the preferred order system, when two or more service resources are available, the customer entity will always pick specific resources over the others. This typically occurs when the customer entity picks the available resource physically closest to the head of the snake queue. This means that in a lightly loaded system, the one or two closest resources will have much higher utilization levels than the furthest resources.

When the head of the queue is centered with respect to the multiple resources, the customer entities may be less prone to pick particular service resources. In this case, the customer entity process for selecting resources may be more random in nature.

Both parallel and snake queues may also exhibit different types of queue behavior. These include:

- Queue priorities
- Queue entity behavior

Queue priority means that the order of the entities in the queue may change according to the priority scheme. These are also sometimes called ranking criterion orders. In any event, there are many different types of queue priorities:

- First In–First Out (FIFO)
- Last In–First Out (LIFO)
- Shortest Processing Time (SPT)
- Longest Processing Time (LPT)
- Lowest Value First (LVF)
- Highest Value First (HVF)
- User-Defined Rules

For most simple systems entities are served on a first-come first-served basis (FIFO). That is, whoever is first in line will be served before later arriving entities. This type of queue priority system is the most commonly encountered in service-type systems. LIFO is another type of queue priority. It is the direct opposite of the FIFO priority. This means that whoever entered the queue last will be the first entity to be processed. The use of LIFO is not nearly as common as FIFO. Most LIFO applications involve some sort of penalty where the less senior members of the queue are faced with some undesirable processing.

Two other queue priority rules that the practitioner may encounter are SPT and LPT. The SPT algorithm can be effectively utilized where there is a service cutoff for the system. Here customers who have the shortest processing time are sent to the head of the queue and are processed first. This type of situation exists in the commercial airline industry. Passengers will arrive at the ticket counter on different flights. It is most important to processes the customers who have flights with imminent departures. If these customers are not given priority, then it is possible that they may miss their flights. Waiting a little extra time in the line will not affect the other passengers with later flights. The LPT algorithm means that customers with the most complicated transactions are handled first. This algorithm is much less common in service systems. It may be encountered only when the longer processing time is of much greater economic benefit to the system. The use of the LPT algorithm will endanger the processing of the customers with shorter processing times. These customers will be prone to reneging.

There is also the LVF priority scheme. LVF priorities are often used to model commercial passenger transportation systems. Passengers can be categorized as first, second, and third class. The value of the class can correspond to 1, 2, and 3. The LVF queue priority would result in first class being at the head of the queue, second class in the middle, and third class at the tail. Second-class passengers can receive service only if there are no first-class customers in the queue. Similarly, third-class passengers can receive service only if there are no first- or second-class passengers in the queue. Any time a first-class passenger arrives in the system, he or she will automatically go to the head of the queue. Similarly, any second-class passenger will automatically enter the queue behind any first-class passenger but before any third-class passenger. The LVF queue priority may also be used to model systems that use a service ticket. Here, customers are served in the order of the lowest ticket. This is similar to the FIFO priority, except there may be customers who temporarily leave the area and then come back for service.

The last normal queue priority method is the HVF priority scheme. Here, each customer has a value assigned that will reorder the position of the customers in the queue in descending order of the value. This type of queue priority might be used when certain individuals are repeat customers. In this situation, the service system may want to provide priority handling to those customers who have done the most business in the past.

It is possible that none of the normal queue priority or criterion-ranking rules adequately models the type of queue priority that the actual system utilizes. In the event that the actual system does use a complicated queue priority rule, most simulation languages provide the opportunity to program in any calculations that are required. These are typically known as user-defined rules.

Queue entity behavior involves the actions of the entities with respect to entering and remaining in the system queues. There are three different types of queue entity behaviors that the practitioner should be familiar with. These are:

1. Balking
2. Reneging
3. Jockeying

Balking occurs when a customer enters the system but leaves before entering a queue. Balking is the result of facing a long queue wait or limited queue capacity. If the customer believes that there are so many individuals in line that an unacceptably long wait will ensue, the customer may depart before getting in line. The assessment of when to balk when facing a long queue wait is entirely individually determined. No two customers can be expected to balk the same. This means that this type of balking must be modeled with a probabilistic distribution. On the other hand, limited queue capacity is usually associated with space constraints. If the length of the queue exceeds the physical space available for the system, customers may not be able to wait in the queue and must leave the system. In this case, the decision to balk will be a specific determination based on physical capacity.

A second type of entity behavior in both parallel queues and snake queues is called reneging. Reneging is when an entity enters the line but leaves before being processed. This would correspond to a customer who is tired of waiting in line and leaves. The decision to renege is also an individual decision. No two individuals will wait the same time before reneging. As a result, the length of time before reneging must be modeled with a probabilistic distribution. Modeling reneging can be slightly more complicated than just waiting a period of time before departing. For instance, a customer who is considering reneging will probably delay this decision if he or she is next in line. Thus, a really sophisticated model will monitor both the current queue time and queue position of each entity.

A final type of queue behavior is jockeying. Jockeying is associated only with parallel queues. This is when an entity switches between two different queues. The decision to jockey is usually triggered by the end of a service period with the resource related to the other queue. The end of the service period results in the entity in the queue leaving the queue and engaging the resource. If the other queue still has more entities or the same number of entities, the entity will not jockey. On the other hand, if the other queue has one fewer entity than the current queue, the entity will jockey. This type of jockeying is complicated but can be modeled in a number of different manners. Jockeying may also occur if the entity perceives that the other queue is moving faster. Jockeying based on this type of perception may not necessarily be able to be modeled.

4.4.2.2 Service Processing

Another event that should be modeled in a service-type system is associated with the type of service that the entity is to receive. To illustrate the different components associated with service processing, we examine a few of the most commonly simulated service system processes. The types of service systems are only representative. The different types of service processes within each type of system should be viewed as providing a starting point for similar simulation system models. Particular service systems may be more or less complex than the examples provided here. We examine the following types of service processes:

- Retail service checkout
- Banking service
- Restaurant service
- Airline ticket counter service

4.4.2.2.1 Retail Service Checkout Processes

Retail service checkout processes are the simplest type of service process that the practitioner is likely to encounter. This type of process is widely observable in department stores, discount stores, specialty stores, and grocery stores. These types of processes simply consist of:

- Calculating the cost of goods
- Payment

Note how the single overall process of buying goods is broken down into two separate service processes. The reason for this is that these two different activities can follow completely independent servicing times. The calculation time for the goods purchased would be expected to be a function of:

- Number of goods purchased
- Type of goods purchased

The greater the number of goods purchased, the longer the process would be expected to take. However, if on the average the goods are large heavy items, the calculation times for the goods may be much greater than those for smaller, more easily handled goods.

For the payment process, it may take just as long to pay for one item as it does for many items. As we see in a following section of this chapter, the service time for payment processes follows its own type of distribution according to whether the customer pays by:

- Cash
- Check
- Credit
- Debit
- Account

As a result, the most realistic type of model will break down these two processes into distinct components. This will increase the probability that our model represents reality.

4.4.2.2.2 Banking Service Processes

In a service-type system such as a bank, there are actually many different types of transactions that the customer may require. For example, for customers seeing the tellers, we might expect:

- Deposits
- Withdrawals
- Money orders
- Cashier's checks

There is likely to be a separate service time distribution for each of these different types of transactions. This means that the practitioner must first determine which types of transactions are to be modeled. The service time delay must be modeled so that it corresponds to the specific type of transaction that is taking place. Here we are not including other types of customers who will also be using other bank services for:

- Opening new accounts
- Closing current accounts
- Safety deposit box transactions
- Mortgages

Obviously, each of these different types of transactions will also require additional modeling and data collection and analysis. The major point in identifying these different types of customers and processing events is that even the relatively simple operation of modeling a bank can become complicated.

4.4.2.2.3 Restaurant Service Processes

Consider a second type of common service system such a restaurant or deli. In this type of service processing we have:

- Placing an order
- Waiting for the order
- Consuming the order

- Augmenting the order
- Paying for the order

In the first part of the process we will be placing an order. The complexity of the order-placing process can depend on whether or not the group orders drinks, appetizers, or main meals. The service processing time may also be dependent on the number of people that are in the group.

Once the order is placed, there will be a probabilistic waiting time for the main order to arrive. During this period, if the restaurant is a sit-down type restaurant, the group may consume their drinks, appetizers, or salads. The length of time that the waiting period takes can be a function of the level of staffing and the quantity of customers already in the restaurant.

Time for consuming the order can vary widely. Some customers will just consume the food while others will include other activities. These include business transactions or discussions or social conversations. The length of time to consume the order may also be a function of the size of the group and the composition of the group. For example, a large family with young children might be expected to take more time than an individual person.

The group may decide to augment the order with after-dinner drinks or dessert. If the group augments the order, the overall service processing time can increase dramatically. Time will be needed for processing the augmented order and consuming the additional food or drink.

Last, the service process will have to include payment for the food and drink. This part of the service process must also include getting the server's attention to calculate the bill and the server's time to generate the bill. As we will discuss later, the overall time for the actual payment can also vary according to the type of payment that the group elects to use.

Note that in some restaurant systems, more than one type of activity may be occurring at the same time. For example in a deli model or a fast-food drive-through, the following activities may be happening simultaneously:

- Waiting for the order
- Paying for the order

In these types of situations, the order is placed and is immediately processed. During the order waiting time, the customer is also either waiting to pay or is actually paying. This has the effect of compressing the overall time that the entire service processing time would have taken if the individual processes were performed sequentially.

4.4.2.2.4 Airline Ticket Counter Service Processing

Airline ticket counter service processing is another representative type of service process. In an airport ticketing system you might expect the following different types of transactions:

- Purchasing tickets
- Checking in
- Changing tickets
- Changing seats

The process of purchasing the tickets would include:

- Determining a suitable flight itinerary
- Payment
- Issuing the tickets
- Checking in luggage

The service delay for determining a suitable flight itinerary would obviously be a function of the complexity of the travel connections with respect to airports and air carriers. The payment would be subject to the same variations as described in the payment service process section of this chapter. Issuing

tickets could be expected to be a relatively short and consistent service period. Lastly, the checking in of luggage would be dependent on the type and number of pieces of check-in luggage.

The process of checking in would be similar to the purchasing ticket process, but include only:

- Issuing tickets
- Checking in luggage

The process of changing tickets could be the result of missing a flight connection. This process would be expected to include:

- Determining the new flight itinerary
- Canceling the old ticket
- Issuing the tickets
- Rerouting baggage

Since this type of transaction is performed in real time at the last minute, the service time for determining the new flight itinerary might take a different service distribution than that of purchasing a new ticket. Canceling the old ticket should require minimal processing time as will issuing the new tickets. Rerouting the baggage, however, may require a different service time than any other process previously described.

The service process of changing seats would be expected to be relatively simple. Here the ticket agent would simply be checking for the availability of other seats. However, two different outcomes could occur. These are:

1. Successful seat change
2. Unsuccessful seat change

The whole process will take a minimum amount of time to check for the seat availability. If the seat change is unsuccessful, then no more processing time is required. However, if the seat change is successful, then the ticket agent will have to reissue a ticket and or boarding pass. This will require additional processing time.

Again, these types of transactions would be expected to have completely different distributions for each type of transaction. Some customers might actually want to perform different types of transactions at the same time.

In the bank, airline, and restaurant examples, it would be generally wrong to assume that there is no difference between service types and to attempt to model the service transaction as a single type with the same service time distribution. While there are cases where it may be necessary to make this assumption, it can lead to other problems, the most serious of which is the inability to be able to state that the model is valid or represents reality.

4.4.2.3 Payment for the Goods or Services

Another system event that the practitioner must observe closely and model carefully is the process of paying for goods or services. This process can be considered as independent of the service processing time because it takes just as long to pay for one item as it does many items. Here the customer's payment options are likely to be:

- Cash
- Check
- Credit
- Debit
- Account

The difference in the time required for these different types of payments can be significant. We would expect that the cash transaction be the fastest, while the check transaction would be the slowest. Credit, debit, and account-type transactions would normally fall somewhere in the middle. However, we must also account for extraordinarily long credit and debit transactions if there is some sort of communications error. The use of probability distributions will assist us in this process. There are distributions with very few observations with long values.

Note that if the payment time is large in comparison to the service-processing time, the payment time can have an overwhelming influence on the overall service time. As with the service-processing event, the practitioner will first have to determine what type of payment transaction will be utilized. A separate payment transaction time distribution will have to be modeled for each type of payment transaction. Again, the consequences of assuming that the payment transaction time distribution is the same for all types of payment can prevent the model from being validated.

4.4.3 Manufacturing System Processes and Events

Let us now turn our attention to manufacturing-type systems. In a manufacturing system the entities are likely to be considered as work in progress or products. The product entities need to be processed as work orders, enter machine queues, and undergo processing times and movement. In addition, machine reliability and product inspection processes must also be modeled. Thus, the practitioner would be interested in including the following components:

- Types of job orders
- Machine queue behavior
- Machine processing
- Machine buffers
- Material transportation
- Machine failures
- Preventive maintenance
- Product inspection failures

4.4.3.1 Types of Job Orders

In a manufacturing system, there are likely to be different types of products that are going to be produced. The products can vary widely in terms of:

- Raw materials
- Components
- Manufacturing processes

It would be expected that these differences would result in different processing times and paths through the manufacturing system. The practitioner will have to determine what parameters exist that differentiate each different type of job order that flows through the system. The differences between the types of jobs can be recorded as a set of entity attributes or a sequence associated with each particular type of job. Sometimes it will also be necessary to include an attribute that uniquely identifies each particular end product. This may be useful in the event that a number of different components for the product will have to be assembled in a later section of the model.

4.4.3.2 Machine Queue Behavior

Just as customers must wait in queues for service, products may have to wait in queues for machine processing. The difference is that a product may ultimately need to enter a large number of queues in a manufacturing system because the manufacturing process for the product may consist of several steps before the product is complete. Each time the product completes one step, it may have to wait for processing for the next step.

In general, most of the types of queue priority or ranking criterion schemes that were discussed in the service queue section are also applicable to manufacturing queues. There are two notable exceptions. In some job order systems, priority algorithms may be based on total processing time or remaining process time. If shortest processing time (SPT) is used as a priority scheme, rapid turnarounds are not held in the queue for unreasonable amounts of time. This can keep up a steady flow of completed jobs.

Another way that the shortest processing time may be utilized in a manufacturing system is with respect to products that are subject to spoilage. The manufacturing system will want to process the products that have the least shelf time remaining before spoiling. This type of queue priority is consistent with the total quality management concept of just-in-time. This concept attempts to minimize the work in progress so storage issues do not become critical.

The second queue exception is associated with manufacturing systems that require a minimum amount of time between processes. This might include a system in which a housing is finished with paint or a resin needs to cure. The longer the time allowed for the paint to dry or the resin to cure, the fewer subsequent finishing or structural defects will result. In this case, the queue priority system would be highest-value-first (HVF). The value would be the amount of time since the last process was completed. If an insufficient interval has passed, the next process may experience what is known as starvation. This is where the process could be performed but cannot because of lack of suitable incoming work in progress.

4.4.3.3 Machine Processing

Perhaps the most important processes to model in manufacturing systems are the machines. Machine processing involves the sequence of machines needed by the product entity. Just like service resources, machines can be laid out in:

- Parallel
- Serial

4.4.3.3.1 Parallel Manufacturing Systems

When laid out in parallel, the product entity can choose from one of the parallel machine sequences. Many large mass production systems are set up in this manner. Sometimes mass production chains are set up in discrete sequences. This means that a particular work in progress goes though one parallel sequence. When it goes through the next parallel sequence, it may shift lines according to the sequence loading and availability. It is also possible that the manufacturing process has some portions laid out in parallel while others are sequential. When a manufacturing system is laid out in a parallel fashion, the resource selection process can become a significant modeling issue. When the entity may select from more than one resource, the selection process must be observed and modeled accordingly. Some of the most common resource selection methods include:

- Preferred order
- Cyclic
- Smallest remaining capacity
- Largest remaining capacity
- Random

When one of the parallel machines has a greater capacity or processing speed, the manufacturing system manager may want to keep the utilization of that machine at a higher level than that of the other machines. Although this will normally increase system throughput, it will cause extra wear on the primary machine. If this is acceptable and actually practiced by the manufacturing system manager, the practitioner will have to use a preferred order approach to modeling the set of parallel machines. This means that the system will always attempt to use the preferred machine unless it is already busy. In that case, the next most preferred machine will be used, and so on.

Sometimes, the system manager does not want any one particular machine among several to receive an inordinate amount of wear. By equally rotating the use of each machine, the system manager can

ensure that each machine receives approximately the same utilization. When this is the case, the practitioner should model the parallel resources using a cyclic resource selection rule.

Some manufacturing system components strive to collect a certain sized batch before the process can begin. This is usually present in a manufacturing process with a large per-cycle setup cost. Here, the objective is to reduce the number of total cycles by making each batch as close to capacity as possible before beginning the cycle. Manufacturing processes that operate in this manner utilize a smallest remaining capacity resource selection rule. As with the actual system, a model that utilizes this rule will try to send the work in progress to the machine that is loaded closest to the specified operating batch size.

The opposite of the smallest remaining capacity resource selection rule is the largest remaining capacity resource selection rule. The use of this rule appears to be much less prevalent than the smallest remaining capacity rule. This rule can be found when the parallel resources each have a large excess capacity and it is better not to have to switch continuously between resources. This situation can be found in process-type industries where it is important to maintain the integrity of individual batches in separate tanks or hoppers.

Occasionally, there does not seem to be any rational method to determining how the work in progress is assigned to a particular parallel machine. In this case, the practitioner can utilize a random resource selection rule to model the system accurately.

4.4.3.3.2 Serial Manufacturing Systems

When laid out in a serial manner, the product entity has no choice but to proceed through the sequence of machines. In the event that the serial layout experiences machine failures, the entire process upstream from the failed machine can come to a complete halt.

The machines in the manufacturing sequence may differ in terms of:

- Capacity
- Processing speed

Machine capacity refers to the number of different work-in-progress entities that the machine can process at a given time. Many machining operations that use computer numerically controlled (CNC) machines can only process one part at a time. However, the single machine can perform many different types of sequential machining operations. On the other hand, it is possible for a machine like a paint sprayer or a curing oven to process many different parts at the same time. One of the common types of simulation studies is to determine if the capacity of a particular manufacturing operation can have an effect on either reducing process bottlenecks or increasing the process throughput.

Manufacturing machines can also vary in processing speed or rate. This is how fast a particular machine can perform its operation. For example, in the tire industry, many types of machines are rated by the number of tires the equipment can produce per hour. With processing speeds or rates, faster is generally better than slower. Sometimes in a simulation model, it is easier to describe the capability of manufacturing machines in terms of processing time or cycle time. This is how long it takes an individual machine to perform one cycle. If you know the processing speed or rate, the processing time or cycle time can be easily calculated. Here smaller values are generally better than larger values. The smaller the processing time or cycle time, the larger is the number of products that can be produced in a given amount of time. This means that if there are a number of different parallel machines in the manufacturing system, the practitioner can model the machines the same way only if the capability of the machines is identical. If the performance of the machines is different because of age or features, the machines must be modeled independently even though they may be positioned in parallel.

The accurate modeling of machines may require not only an evaluation of the machine's capability. In any type of manufacturing system where there is a human operator, the practitioner must also account for the operator's effect on the machine's performance. A less-capable machine with an expert operator could easily equal the performance of a much higher-capacity machine with a less-experienced operator.

The simulation study could easily indicate that a more capable machine is needed when, in reality, the operator just needs to be trained better.

4.4.3.4 Machine Buffers

As we previously discussed, a manufacturing system may consist of a number of individual manufacturing processes. The product entity will flow between the processes. In many systems of this type, there is a limited amount of space between the individual processing machines. In this case, the limited queue capacity would block the forward movement or further processing of the work-in-progress parts. This process is known as blocking. Blocking can occur both upstream and downstream of a particular process.

4.4.3.5 Material Movement

In a large manufacturing facility, some transportation time may be needed to be modeled. This may be in the form of conveyor or transporter movement.

4.4.3.5.1 Material Movement by Transporter

The actual process of movement by a transporter in a manufacturing system may include the following processes:

- Waiting until the transporter arrives at the work-in-progess's location
- Loading the work in progress onto the transporter
- Actual transportation process
- Unloading from the transporter

In the component section of this chapter, we identified transporters as forklifts, dolleys, and vehicles. When these types of transporters are used to move a work in progress between locations, the work in progress must first make a request to be transported. In the actual system, this might consist of a batch of components completed in one part of the manufacturing plant. The completion of the batch means that the work in progress must be transported to either another process or a storage area. Either way, the batch will remain in its current position until the appropriate transporter is available.

When the transporter is available, there will ordinarily be some sort of delay while the work in progress is loaded onto the transporter. During this period of time, the transporter is not available for any other tasks and must also normally remain stationary. In addition to the transporter, there may also be other types of material-handling equipment such as a crane and also possibly operators involved in the process. The larger, heavier, more numerous, and more difficult to handle the work in progress, the longer the loading time would be expected to be. At the completion of the loading process, the other material-handling equipment and the operators if any will be released for other activities.

When the transporter is ready to move, the practitioner must determine all of the possible end locations for the work in progress. The work in progress may go to one of any number of other machine locations according to the work process or to a storage location. The distances and or travel times between these different possible locations must be determined as well.

When the transporter ultimately arrives at the proper location, there will be another service delay to model the unloading process. As with the loading process, the unloading process may also require the use of additional material-handling equipment as well as human operators. Again, the larger, heavier, more numerous, and more difficult to handle the work in progress, the longer the unloading time would be expected to be. Once the unloading process is complete, the other material-handling equipment and the human operators may be released for other transportation requests.

4.4.3.5.2 Material Movement by Conveyor

The actual process of movement by a conveyor in a manufacturing system may include the following processes:

- Waiting to access the conveyor
- Loading the conveyor

- Actual transportation process
- Unloading from the conveyor

In a heavily loaded conveyor system, there may be some delay before the work in progress is able to access the conveyor. It could turn out that the conveyor is temporarily not moving because an unloading process is taking place, or it could be that there is no space on the conveyor. Either way, until the work in progress is able to access the conveyor, there may be a delay with the work in progress remaining in its current position.

When the conveyor is operating and there is space for the work in progress, there may be an additional delay for the work in progress to be loaded onto the conveyor. As with the transporters, the loading process for a conveyor may also require some sort of material-handling equipment as well as human operators. A typical type of material-handling equipment for a conveyor is a small robot arm. The arm may remove work in progress from a pallet or other temporary storage onto the conveyor. In a similar manner as with transporters, the heavier, bulkier, and more difficult to handle the work in progress is, the longer the conveyor-loading time may be.

Once on the conveyor, the work in progress will be moved to its final destination. In many cases, the conveyor will move the work in progress to a specific location rather than to one of several locations as with transporters. When modeling transporters, it is necessary to measure the distance between the conveyor loading and unloading areas. It is also necessary to record the movement velocity of the conveyor during operation.

When the work in progress arrives at the unloading area, there may also be some delay while the work in progress is unloaded. As with the loading process, the work in progress may require material-handling equipment and a human operator to be available. Once all of the necessary unloading resources are available, the work in progress can actually be unloaded. Our heavier, bulkier, more difficult to handle issues must also be taken into consideration when modeling the unloading process.

Failure to model the overall transportation time will make the model appear as though the product is moving instantaneously between areas. This can have a negative effect on the overall system time and possibly prevent model validation.

A final conveyor issue involves the size of the work in progress with respect to the actual conveyor, not the material-handling equipment. In instances where differently sized objects are placed on the conveyor, it is possible that an insufficient amount of space will be available on the conveyor belt when the work in progress is to be loaded on the conveyor. The situation might actually occur that a subsequent piece of work in progress could be loaded onto the conveyor instead. Thus, subsequent smaller work-in-progress entities are blocked by the large first-in-line work-in-progress entities. If, for example, a human being is involved in the system, it is possible to circumvent these types of blockages. If the actual system has this sort of capability, the practitioner will also have to model this logic. The resulting model can become complex. These types of situations can occur in shipping-type industries that handle different sizes and shapes of packages. It can also occur in airport baggage-handling systems.

4.4.3.6 Machine Failures

Anytime there is a machine involved in a manufacturing process, there is the possibility of some sort of machine failure. The machine failure will prevent any additional processing to the product currently being produced. In addition, no other product entities will be allowed to be processed until the machine failure is resolved. These machine failures can include:

- Broken components
- Jammed machines

The practitioner will have to identify a list of common machine failures resulting from broken components. Each of these different types of machine failures can have different time distributions to repair.

Some of these broken components will be relatively quick to repair, such as a broken machine tool cutting bit. On the other hand, other machine tool failures may result in the complete shutdown of that machine for an extended period of time. An example of this is a burnt-out circuit board.

A jammed machine will also prevent the machine from processing the product entity. However, in this case, the machine failure could be limited to the removal of the product entity from the machine. The machine may then be able to continue on as before. The product entity that was being processed will be an inspection failure as discussed in the next section.

4.4.3.7 Preventive Maintenance

Preventive maintenance is performed in many manufacturing systems to help reduce the probability of machine failures. Preventive maintenance processes may be required after so many manufacturing cycles or after so many hours of machine times. Other preventive maintenance processes are required on a calendar basis. Generally speaking, preventive maintenance processes will require that the manufacturing operation be temporarily suspended. Preventive maintenance can come in the form of:

- Calibration
- Tooling replacement
- Lubrication
- Cleaning

High-accuracy manufacturing operations will require that the manufacturing equipment be periodically calibrated. This will ensure that the machine settings are set correctly and have not accidentally drifted, or the tooling has not become distorted.

When manufacturing processes include any type of machining operations, tooling replacement processes are regularly performed as preventive maintenance. Manufacturing tooling will wear according to the raw material, cutting feed rates, and the amount of machining time.

Manufacturing equipment generally requires some sort of lubrication in order to function correctly. Proper lubrication will maintain cycle rates and reduce wear on the manufacturing equipment. Proper lubrication can also be required to reduce corrosion.

In many manufacturing operations, it is essential that the manufacturing equipment be cleaned. In the food-processing industry, cleaning may involve the removal of food byproducts. In machining operations, cleaning processes can include the removal of metal chips, ribbons, or slag.

When attempting to model these types of preventive maintenance operations, the practitioner can make use of special preventive maintenance entities. These entities are not physical in the same sense as normal job, order, or product entities. Preventive maintenance entities are more analogous to events that occur during the course of the simulation run. The preventive maintenance entity is created by being either scheduled or generated as some other event occurs, such as the completion of the required number of cycles. When the preventive maintenance entity is created, it will take control over the manufacturing resource. The entity retains control over the manufacturing resource until the end of the preventive maintenance operation. At the end of the preventive maintenance operation, the preventive maintenance entity releases control over the resource. At this point, if the entity is no longer needed, it is discarded.

4.4.3.8 Inspection Failures

In a manufacturing process, the product entity will go through a series of processes that add value to the product. In many processes, it is possible that the product becomes defective in some manner or another. For this reason, the practitioner must determine what types of product inspection failures may exist. These product failures may be either

- Reworked
- Scrapped

In the event that the product can be reworked, the practitioner must determine what additional processing is necessary to be able to insert the product back into the overall manufacturing process. This additional

processing could include additional testing to determine the exact nature of the failure or the disassembly and removal of the defective component.

In the event that the product is to be scrapped, it may be necessary to update a variety of statistical counters or system variables. The use of counters and variables will assist the practitioner in keeping track of manufacturing costs that result from scrapped defects.

4.4.4 Events Not to Model

The practitioner may deliberately elect not to model some events. These would typically include events with very limited impact on the system outputs. Limited impact may be the result of the small importance or infrequent occurrence of the event. Some events that would not normally be included in a system include:

- Discovery of an explosive device in a security checkpoint system
- Power outage in a manufacturing facility
- Bus involvment in an accident
- Workers' strike

4.5 Data to Be Included in the Model

As you are probably already aware from either the introduction chapter or your previous experience, there are input data and output data. Input data are what drive the system, and output data are what result from the system. In this section, we will take a high-level view of the model data requirements. In Chapter 5,"Input Data Collection and Analysis," we cover this subject in greater detail. The subjects covered in this section include:

- Input data
- Input data considerations

4.5.1 Input Data

In the system definition process, the practitioner is interested in identifying a preliminary list of the types of input data that affect the system's output data performance. As the practitioner begins to collect data or model the system, the preliminary list may be modified. In this section, we discuss some data collection principles and identify some of the more common types of input data.

4.5.1.1 Input Data Collection Principles

A fundamental concept in this process is to break down the types of data into as many different independent types as possible. Consider the processing of commercial air passengers through a security checkpoint system. One practitioner created a model that took into account only the total processing time for each customer. Subsequent analysis of this security process indicated that the same processing time actually consisted of several different types of input data. These included, among others:

- Number of carry-on bags
- Time to load each bag on the x-ray machine
- Time to convey and x-ray a bag
- Processing time for the metal detector including emptying pockets
- Pass/fail rate for the metal detector
- Time to remove each bag from the x-ray conveyor

Consider another example with the staffing requirements for a ticket booth at a movie theater. We would not necessarily be interested in the number of tickets sold per hour. We would really be interested in the

number of individuals who buy tickets and how many tickets each individual buys. By breaking the data down in this manner, we can always find the total number of tickets sold. However, if we had only counted the number of tickets, we would not be able to determine the number of individuals who had purchased tickets. It is the number of individuals purchasing tickets that would most affect the staffing requirements.

4.5.1.2 Types of Input Data

There are two general categories of input data. The first category is related to the concept of system entities. Entities can be thought of as the elements that are processed by the system. Common examples of entities include customers, passengers, and job orders. Typical types of input data associated with entities are the time between arrivals to the system, the number of entities in an arrival, and entity processing times for various operations.

The second category involves system resources. System resources are the parts of the system that process the system entities. Common examples of system resources are personnel, manufacturing equipment, and vehicles. System resource input data include break times, breakdown or failure rates, operating capacities, and movement speeds.

In the interest of facilitating the input data collection process, some of the most commonly encountered types of input data are identified in this section. The selection of these common types of input data does not mean that other types of input data will not be encountered. The analyst must be prepared to identify types of input data particular to the specific project.

4.5.1.3 Interarrival Times

Interarrival times are the amount of time that passes between the arrivals of batches or groups of entities into the system. Entities can include customers, orders, jobs, or breakdowns. Data of this type are also commonly found in the form of arrival rates. The interarrival times are the inverse of the arrival rate. For example, if there are five arrivals per minute, the interarrival time is 0.2 min. This is the same as an interarrival time of 12 s.

Interarrival times are normally considered to be probabilistic in nature. Many events follow a random arrival process. In this event, mathematical proofs can demonstrate that the interarrival times follow an exponential distribution. Because this handbook focuses on practitioner needs, readers will be spared this demonstration. Interarrival times are not always probabilistic. In the case of production scheduling, the release of a production order may be decided in advance according to needs.

4.5.1.4 Batch Sizes

Batch sizes involve the number of individual entities that arrive at the same time to be processed by a system. An example of a batch size is the number of airline passengers that are traveling together and arrive at a security checkpoint system. Although the batch arrives at the same time for processing, each individual in the batch must be processed individually. Batch sizes are generally probabilistic and discrete in nature.

4.5.1.5 Balking, Reneging, and Jockeying

As discussed earlier in this chapter, balking, reneging, and jockeying are queue-related behaviors exhibited by system entities. Recall that balking involves the entity's observing the operation of the queue and leaving the system before entering the queue. The decision to balk can be either probabilistic or deterministic. If a customer decides that the line is too long and leaves, a subjective probabilistic decision was involved. Conversely, if the balking is caused by limited space in the line, then the decision to balk would be deterministic. As previously introduced, reneging involves the entity's actually entering the queue but leaving the queue before receiving service. Reneging would normally be considered to be a subjective probabilistic decision. Last, we defined jockeying as the movement of entities between similar queues in the expectation of receiving earlier service. Jockeying would also normally be considered to be a subjective probabilistic decision. However, in actual practice, many individuals will choose to seek the queue that holds the promise of earlier processing.

4.5.1.6 Classifications

Classifications include types or priorities of entities arriving in the system. Classifications are commonly used to determine the process scheduling of jobs or customers. Classifications would normally be expected to be probabilistic in nature.

4.5.1.7 Service Times

Service times include processing times that a job or customer undergoes. This includes time during which the entity occupies the attention of a resource such as a machine, operator, or clerk. It specifically does not include the time during which an entity waits in a queue for processing. Service times may be either probabilistic or deterministic. Probabilistic service times are likely to include the presence of humans. A deterministic service time is more likely to involve some sort of automatic processing such as running a computer numerically controlled machining program.

4.5.1.8 Failure Rates

These rates involve the frequency of process failure or resource unavailability. Failure rates are normally probabilistic in nature. Examples of failure rates involve inspection processes and machine breakdowns. Both of these rates would be expected to follow some sort of probability distribution. In the case of a machine breakdown, there would also likely be a repair time. This type of input data can be considered to be similar to a service time.

4.5.1.9 Scheduled Maintenance

This involves reducing the availability of resources such as machines to perform preventive maintenance to reduce the probability of equipment failures. Scheduled maintenance is normally deterministic in nature. However, the implementation of the scheduled maintenance may be according to calendar time, operating time, or the number of jobs that have been processed.

4.5.1.10 Break Times

Break times primarily pertain to system resources such as operators and clerks. In most cases, these individuals will be unavailable because of breaks and meal periods. It is likely that the duration of the break times will be deterministic. One issue that the practitioner will need to model is exactly how the resources operate when the break occurs. There are three basic methods of modeling breaks.

1. Wait until the end of service
2. Ignore the start of the break
3. Preempt the current service

In the first type of break model, if the resource is busy, the resource will wait until any entity being served has completed its time before going on the break. When the entity does go on break, the duration of the break is as long as it was originally scheduled for. In the event that the resource is not busy, the resource will go on its break immediately for the full duration.

In the second type of break model, if the resource is busy, the resource will continue to service the entity until the end of the service period. At the end of the current service, the resource then goes on break. However, in this type of model, the entity is only entitled to the remaining break time after the beginning of the scheduled break. In the event that the service time exceeds the break, the resource is not allowed to go on break. If the resource is not busy, the resource will go on break immediately for the full duration as with the first type of break model.

In the final type of break model, if the resource is busy, the resource will immediately go on the break. The entity that was being served will be put aside and will wait until the resource comes off the break. When the resource comes off the break, the remaining service time will be continued. If the resource is not busy, the resource will immediately go on break for the full duration.

4.5.1.11 Movement Times

Movement times can include the duration for an entity to travel between areas of a system. This can include movement on foot, by vehicle, or conveyor. The movement times may be deterministic in the case of a conveyor or probabilistic in the case of an individual walking.

4.5.2 Other Input Data Considerations

The importance of collecting accurate and complete input data cannot be overemphasized. However, the practitioner may be interested in a high-level model and may have only limited project time. In this situation, collecting an exhaustive amount of data may not be in the best interest of the practitioner. When this occurs, the practitioner should make note of what input data collection compromises were necessary. These compromises should be clearly documented under the assumptions and limitations section of the project report.

4.6 Output Data

Output data are generally used to measure performance levels. These measures of performance are used to validate the model and conduct model experimentation. In the model validation process, output measures of performance are collected and compared from both the actual system and the simulation model. In model experimentation, we create models with different operating and resource policies. We then compare the output measures of performance between the different models.

4.6.1 Primary Measure of Performance

Practitioners frequently collect multiple types of output measures of performance for a single simulation study. This is not normally a difficult task because of the output data collection abilities of most modern simulation software packages. Although many different types of output measures of performance may be easily collected, the practitioner will still have to decide which measure to use as the principal means of comparison. The decision as to the most appropriate output measure of performance to use can be made only by the practitioner. In some cases, the practitioner can return to the original project objectives. If, for example, the driving force was to reduce operating costs, the practitioner could focus on resource utilization. If resource utilization was low, fewer resources may be needed. Conversely, if customer satisfaction needed to be improved, the practitioner might focus on system time as the primary output measure of performance.

Some of the commonly used output measures of performance and their mathematical definitions were introduced for simple systems in the Introduction (Chapter 1). The following section provides additional information on these measures:

- Average time in the system
- Average time in a queue
- Time average number in queue
- Average utilization rates
- Counters

4.6.1.1 Average System Time

System time is defined as the complete time each entity is in the entire system or a component of the system. It typically includes queue time, service time, and transportation time within the system. Queue time is the length of time the entity waits in a specific queue. Service time is the period during which the entity is utilizing a resource. Transportation time is the length of time that an entity is transported between components within the system. In some cases, the practitioner may choose also to include transportation time to enter the system and transportation time to exit the system. These would typically

be included in facilities design-type simulations. The following formula is used to calculate average system time:

$$\text{Average System Time} = \frac{\sum_{i=1}^{n} T_i}{n}$$

where T_i is the system time for an individual entity (arrival time – departure time), and n is the number of entities that are processed through the system.

4.6.1.2 Average Queue Time

Queue time is defined as the length of time that each entity waits in a particular queue. It begins when the entity enters the queue, and it ends when the entity leaves the queue for processing or transporting by a resource. Average queue time is simply the total time that all entities wait in a particular queue divided by the total number of entities that were processed through the queue. If multiple queues exist in a model, it makes more sense to calculate the averages of each individual queue. This method makes more sense because unless the system is extremely well balanced, particular queues will have longer average queue times than others. If the average queue times are lumped together, any differences between the queues will be lost.

$$\text{Average Queue Time} = \frac{\sum_{i=1}^{n} D_i}{n}$$

where D_i = queue time for an individual entity (queue arrival time – service begin time), and n = number of entities that are processed through the queue.

4.6.1.3 Time-Average Number in Queue

The time-average number in queue is the number of entities that can be expected to be observed in the queue at any given time. Time-average number in queue is frequently confused with the average queue time by novice practitioners. As Chapter 1 illustrated, the time-average number in queue is time dependent, whereas average time in queue is observational:

$$\text{Time Average Number in Q} = \frac{\int_0^T Q\,dt}{T}$$

where

Q = number in the queue for a given length of time

dt = length of time that Q is observed

T = total length of time for the simulation

The time-average number in queue calculations rarely result in discrete numbers. Novice practitioners are also frequently disturbed by the seeming impossibility of observing a fractional entity in the queue at any given time. Only when the practitioner understands what this measure represents do fractional numbers make sense.

When multiple queues within the model for processes are involved, separate time-average number in queue calculations should be made. The only exception to this would be if there were a number of

identical queues serving the same process within the model. In this case, the average of the multiple queue values could be calculated.

4.6.1.4 Average Utilization Rates

The utilization rate for a resource is the percentage of time that the resource was being utilized by an entity. If there is more than one resource for a given process, the practitioner may proceed in one of two manners. If the resources are identical with respect to processing capability, the practitioner can add up all of the average utilization rates then divide by the number of resources. In the other situation where resources are not identical with respect to processing capability it would be better to collect average utilization rate data for each individual resource:

$$\text{Average Resource Utilization} = \frac{\int_0^T B\,dt}{T}$$

where

$B = 0$ for idle or 1 for busy

dt = length of time that B is observed

T = total length of time for the simulation

4.6.1.5 Average Utilization Rates for Multiple Resources

Sometimes the practitioner will choose to model several parallel identical system resources with one model resource with a capacity larger than 1. The advantage of this is a far more simplistic model. As long as the capabilities of the different parallel system resources are actually identical, this simplifying modeling technique can be utilized. However, when using this approach, the practitioner must realize that the model statistics associated with the average utilization rate must also be properly translated. This means that the overall utilization values must be divided by the capacity of the single multiple-capacity resource.

For example, if there are two identical resources modeled as a single resource with a capacity of 2, the instantaneous utilization value will be between 0 and 2. This means that any one of the following situations may be occurring:

- Both resources are idle.
- One resource is busy and the other is not.
- Both resources are busy.

In order to translate properly the resource utilization statistics, the practitioner must insure that the values are divided by 2. This means that the instantaneous value will now fluctuate as either 0, 0.5, or 1. Similarly, the average utilization value will now take a value between 0 and 1. These numbers can actually be misleading because one of the two resources may be doing the bulk of the work. If the values are consistently around 0.5, it could mean that one resource is almost always busy while the other resource is almost always idle.

4.6.2 Counters

Counters can be used to keep track of the number of entities that have been processed through either the entire system or a particular component of the system. Counters can be either incremented or decremented. An example of an incrementing counter would be the number of customers entering a store. An example of a decrementing counter would be the number of seats available on a bus. Counter increments and decrements are normally one unit. However, in some cases multiple-step values can be used when entities represent more than one unit. Multiple step values would be appropriate to calculate the number of individual components in a case of components.

In addition to performance purposes, counters are also frequently used for verification purposes. In this case, the practitioner would use a counter to ensure that entities are processed through particular parts of the system. If entities should be coming through a component of the system, but no entities are counted, the practitioner will need to investigate the model for logic errors.

4.7 Summary

In this chapter we discussed the procedure to define the system that we are interested in modeling. This process began with classifying the type of system. Systems can be discrete, continuous, or combined with respect to the system events. One way of determining the type of system is by examining the type of entities that flow through the system. Entities like people automatically make the system discrete. Entities that are fluids or act as fluids are generally continuous event simulations. Entities that start out as fluids and end up as discrete units are combined systems.

Another way that systems are classified is as terminating or nonterminating. Terminating systems have an event that ends the time period of interest. In general, terminating systems are also cleared of entities at the end of the time period. In contrast, nonterminating systems do not close, or, if they do close, the entities remain in the system until the system reopens. The difference between terminating and nonterminating systems is significant because each type of system requires a different type of analysis approach.

Another important component of the system definition process is to decide what components and events to model. The practitioner must make decisions on how wide a scope the model should cover and how great the model detail should be. The practitioner also needs to determine what type of input data needs to be collected and analyzed. Finally, the practitioner must decide what sort of output data needs to be generated. These output data will eventually be used to make statistical comparisons between model alternatives. At the end of the system definition process, the practitioner can begin collecting input data and translating the model into a simulation programming language.

Chapter Problems

1. What differentiates a terminating system from a nonterminating system?

2. What is the difference between a discrete and a continuous event system?

3. Identify five different types of input data for an airport security checkpoint system?