Contents

Chapter 6 Building Basic Simulation Models

The Intermediate User uses simulation as part of his or her job description, but it is not usually the main focus. Typical job titles of the Intermediate User include operations analyst or industrial, process, packaging, or manufacturing engineer. Individuals engaged in manufacturing research and development may also be Intermediate Users. Typically, the Intermediate User will be involved with simulation five or six times per year. Intermediate Users can build relatively straightforward simulation models using drop-down menus or wizards. When more detailed simulations are required, the Intermediate User can call on technical support or third-party consultants.

Using simulation as a basic tool can be accomplished with the use of pre-built logic and a minimal amount of training.

This chapter introduces the basic structure and components found in most simulation software modeling and analysis packages. It includes a discussion of the modeling environment, introduces the basic structure and functionality of modeling objects, defines relationships among objects, describes how to move items through a model, and discusses obtaining output statistics from a single run of a model. This chapter will illustrate the object functionality as used in the *FlexSim* application. The remainder of the book builds the reader's capability to create and analyze simulation models through the *FlexSim* software. While *FlexSim* is used in this explanation, the functionality is similar to other applications, but is implemented differently.

This chapter, and those that follow, contains exercises that will develop those simulation skills used by Intermediate and Advanced Users. These skills, which are needed to build simulations, are discussed in general terms in the main chapters, while specific details of the *FlexSim* implementation are contained in the Appendix.

The discussions in this chapter are meant to be a quick-start introduction to simulation modeling. Some of the topics will be explored in greater detail in later chapters. Consult the Appendix as well as the Users' Manual in the Help section of *FlexSim* for further information about using the software.

Section 6-1 Simulation environment

FlexSim, like other professional simulation software, provides extensive support for building and analyzing simulation models. Figure 6.1 outlines the basic components found in such software. Most of the components, especially those in the left portion of the figure, will be discussed later in the book.

The simulation environment for most software applications is similar and follow generally accepted conventions.

This chapter concerns the modeling environment and focuses on the components in the right portion of Figure 6.1. All simulation software provides a set of pre-defined modeling objects that facilitates model building. Such objects include queues that store items awaiting processing (due to unplanned delays), processing objects that modify or create planned delays for items as they flow through the model (service operations), transporting objects that move items through the model, etc. The number and capability of such objects varies widely in the various simulation software products that are on the market today. Some software, like *FlexSim*, allows users to easily change the behavior of objects, and Advanced Users can create their own objects.

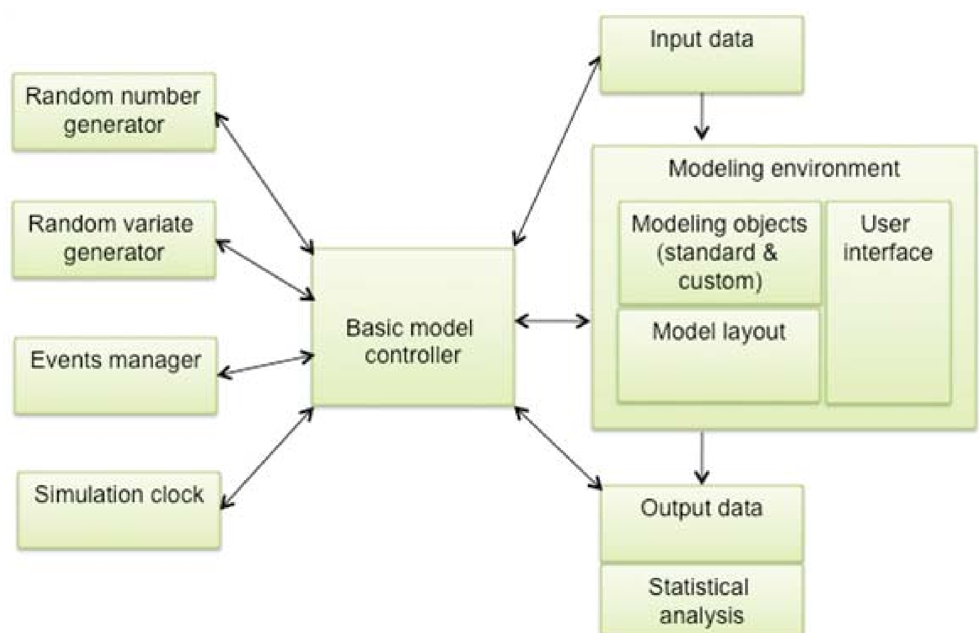


Figure 6.1 Basic components of simulation software

The modeling objects are typically placed and arranged on a layout, or simulation surface, that resembles the physical layout of the system being modeled. The objects are connected to represent the flow of items through the model and permit communication among objects.

All simulation software products have their own user interface that enables users to effectively and efficiently build models, import data, analyze results, etc. While much of the underlying logic is the same across the various simulation software products, the user interface and accessibility to the underlying logic varies extensively. *FlexSim*'s interface is quite effective for all levels of users; however, much of its power lies in its openness and ability to modify the objects and their behavior to best match the modeling requirements.

As mentioned above and shown in Figure 6.1, a simulation model is data driven; that is, without data, the model is useless and cannot function. Simulation software, therefore, needs to provide mechanisms to easily import data. Similarly, the reason we build and execute models is to obtain information and to understand the consequences of actions; therefore, simulation software needs to provide mechanisms that easily present or export data and that facilitate the analysis of that data.

Simulation environment terminology

When describing a simulation, certain terms are used to talk about the component pieces. While the specific names may vary in each software application, the concepts and functionality are basically the same.

Time and space

Time and space in the simulation environment are dimensionless until the user decides what the dimensions should be. For some reports, *FlexSim* provides a default set of units: Seconds, Meters, and Liters. To change the defaults go to the File/ Global Preferences/ Environment and select the desired units. *FlexSim* then has to be closed and re-opened for the change to take place. In the same place you can choose to have the screen displayed whenever a new model is created.

When starting a new model, a screen appears as shown in Figure 6.2, showing the definition of units. The values are used during some reporting and internal calculations. All data input for the model should be consistent with these units. For example, if the time unit is seconds, the time between arrivals and cycle time for a processor would both have to be expressed in seconds. As a result, statistics that are generated by the simulation, such as the amount of time a processor is operating, will be interpreted in seconds. Once selected, the units may not be changed until a new model is opened. The units for a model are saved with the model file.

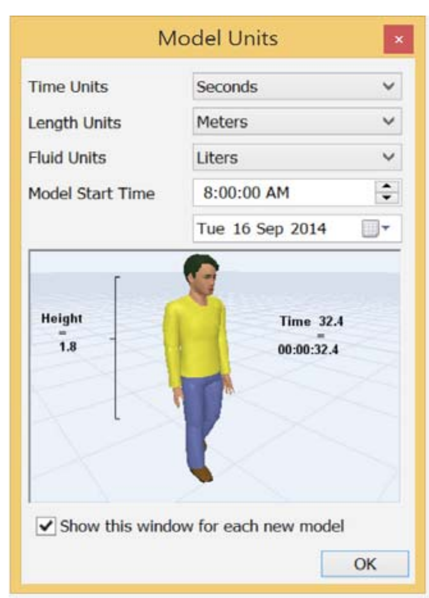


Figure 6.2 Units screen

Space is also dimensionless. The default unit of space on the simulation layout screen is one grid unit of size 1x1x1. The user has to decide the actual units that the default grid space represents. Equipment can be scaled appropriately. Task executers, such as fork trucks, have speed values assigned to them as the number of grid units traveled per clock unit. Being consistent with data values is critical to a successful simulation.

The main screen

Chapter 3 dealt with models that were already built along with the simulation environment to run them. This section starts with a blank page or layout known as the main screen as shown in Figure 6.3.

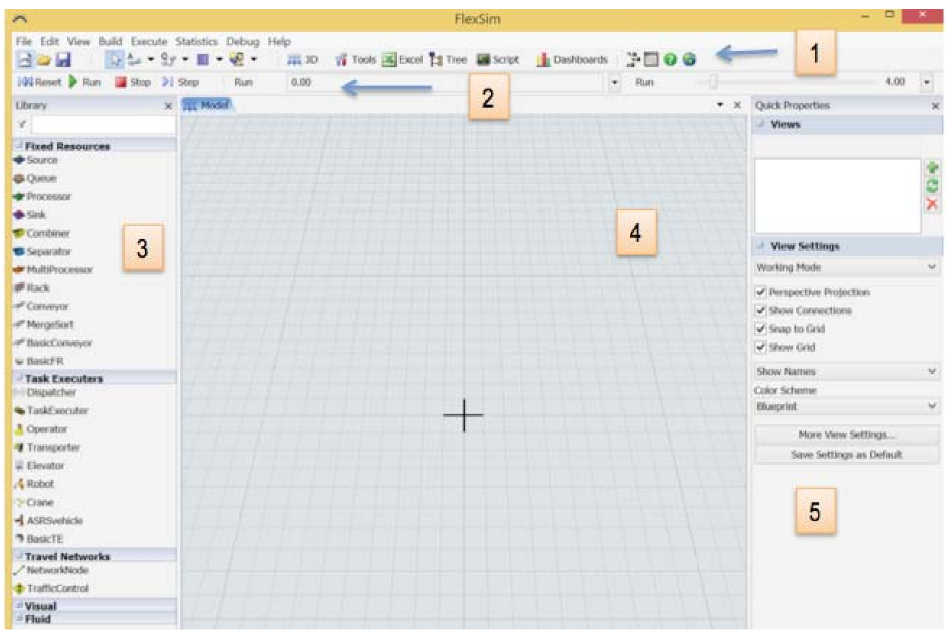


Figure 6.3 *FlexSim* model development main screen interface

As discussed in the previous section, all simulation applications have an environment in which a simulation model is built. That environment contains the surface for building the model (sometimes referred to as a layout), menus for controlling files and functions, and basic units or objects that are used to build the simulation model. *FlexSim*, as with most other simulation software, uses the conventions associated with Microsoft applications (file management, drop-down menus, tool bars, etc). As a side note, in the student version of *FlexSim*, the watermark "Educational Version" appears on the layout.

The main screen is made up of five major areas:

Area 1: Main toolbars

- Top row: Set of Microsoft Windows style drop-down menu options for file control as well as *FlexSim* operating controls
- Lower row: Shortcut buttons that allow quick access to some common *FlexSim* interfaces, operating elements, and model views

Area 2: Simulation control panel

- Buttons
    - **Reset:** Initializes the model
    - **Run:** Begins execution of the simulation
    - **Stop:** Stops the model at the end of the current clock cycle (A simulation can be re-started from the stop point.)
    - **Step:** Moves simulation ahead to the next scheduled model event
- Time controls
    - **Run Time:** Displays the current model time in simulation time units
    - **Stop Time drop down arrow:** Sets a specific time at which the simulation will be stopped (Once stopped, the value can be changed to set a new future stop time)
    - **Speed Slider:** Defines the number of simulation time units per second of real time. The drop down arrow can set a specific rate.

Area 3: Library icon grid

- Objects that can be brought into the simulation model by clicking and holding down the mouse button while dragging the object onto the modeling surface or layout. Other libraries can be added.

Area 4: Model view window

- Surface where the simulation model is built

Area 5: Quick Properties pan

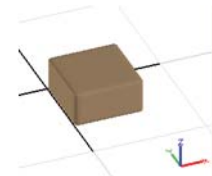
- Displays that can be used to monitor and change data for various simulation objects. The displays will change based on the particular context.

Most applications allow modifications to the basic screen. The Appendix shows how the *FlexSim* simulation workspace can be visually changed.

Section 6-2 Simulation components

Graphically-oriented simulation software applications make use of elements that are placed on the simulation surface and can be viewed. In *FlexSim*, an **object** is the most basic building element of a simulation.

In *FlexSim* there are two basic types of objects: **discrete** and **fluid** (continuous). The discrete objects are used to develop discrete-event simulation models where model behavior results from events that occur at discrete points in time, such as an item arriving to the system or a machine stopping due to an internal failure. Fluid or continuous objects are used to model behavior that results from changes that occur continuously over time, such as the filling of a tank with a liquid. Most of the focus of this book is on discrete-event models and the use of discrete objects. Fluid objects are discussed in Chapter 14.



Simulation objects perform specific functions while flowitems move within the simulation.

Simulations normally involve actual discrete entities that physically move around in the simulated environment. In *FlexSim*, these entities are called **flowitems**. Depending on the simulation, these could be boxes, products, customers, paperwork, and so on. Without flowitems, there isn't a need for simulation. Some simulation software packages refer to flowitems as entities or transactions.

In *FlexSim*, flowitems are listed in the Flowitem Bin which is accessed through the Tools menu at the top of the screen. Normally, flowitems are brought into a simulation through the source object where the choice of using a 3D flowitem is listed in a drop-down menu. Additional details are included in the Appendix.

Simulations also need objects that interact with flowitems. These might perform an operation, create a delay, or move the items. In *FlexSim*, there are two general categories of such discrete objects: fixed resources and task executors (mobile resources).

A **simulation model** is simply a collection of these objects put together in such a manner as to simulate the behavior of a system. The *FlexSim* simulation model shown in Figure 6.4 is one such combination of objects.

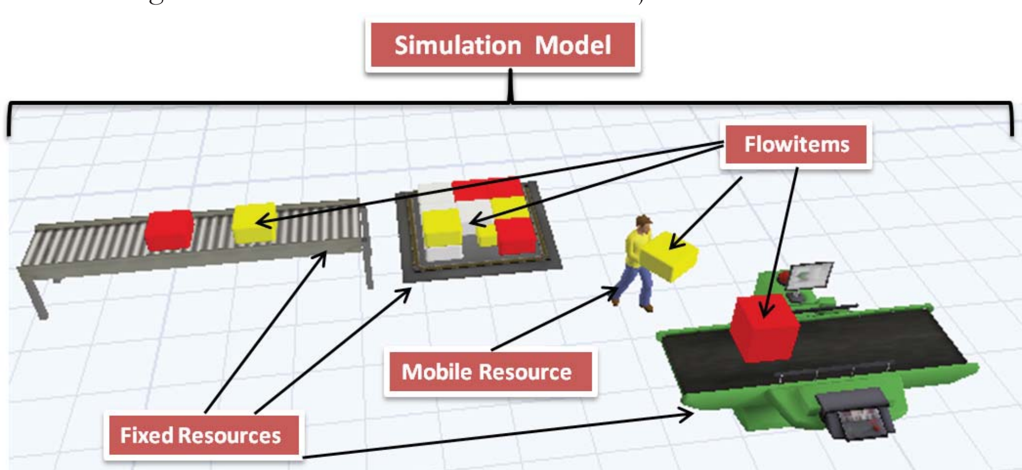


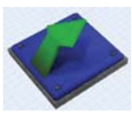
Figure 6.4 ***FlexSim*** **simulation terminology**

Section 6-3 Fixed resources

Fixed resources are the objects that send, receive, and perform activities/operations on flowitems. They are also the most common object type. They are referred to as "fixed" because they are largely stationary. Once they are placed on the model surface, they tend to stay in that place unless manipulated later by the modeler. Examples of this type of object include the following:

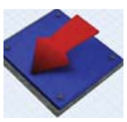
Objects that normally stay in one position and perform operations on a flowitem are called fixed resources.

Source



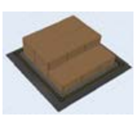
- Creates and releases flowitems
- Modes of arrival: interarrival time, arrival schedule, arrival sequence

Sink



- Receives and removes flowitems from the simulation

Queue



- Temporarily stores flowitems when downstream objects cannot accept them
- Can receive multiple flowitems at a time
- Can batch process flowitems
- Receives flowitems until its specified maximum content is reached

Processor

- Processes or forces a delay on a flowitem
- May call operators for setup or process operations
- May incur scheduled or unscheduled downtimes
- Handles one flowitem at a time or multiple flowitems independently (if its maximum content > 1)

MultiProcessor

- Performs a set of operations or processes in sequence
- Operations/processes may have separate times and call separate resources
- Handles one flowitem at a time

Conveyor

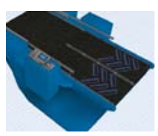
- Moves flowitems over a fixed path (not necessarily linear) at a specified speed
- Flowitems enter and leave the conveyor one at a time
- Modes: accumulating, non-accumulating

- Capacity limited by number of flowitems or available space on the conveyor. Spacing between flowitems may be specified.
- The library may contain many types of conveyors that allow advanced functions such as defining when and how flow items leave the object.

Combiner

- Groups multiple flowitems
- One input from Port 1 which may be a container to hold flowitems from other ports
- Can have process times, use resources, and incur scheduled or unscheduled downtimes
- Modes: join, cannot be separated; pack, container from port 1 holds others—can be separated; and batch, all objects travel as a group.

Separator



- Accepts one input at a time.
- Can have process times, use resources, and incur scheduled or unscheduled downtimes.
- Modes: unpack from packed container or split, makes copies of a flowitem.

Section 6-4 Transporting items

The transportation of flowitems in a *FlexSim* model is the physical movement from one resource to another. That movement may occur in a number of different ways. By default, fixed resource objects in a model pass flowitems to each other instantly. In addition, flowitems may move from one fixed resource to another through the use of an intermediary fixed resource object, such as a conveyor.

Task executers move flowitems from one place to another.

Flowitems may also be moved by **task executers**. These are special objects that have mobility within a model. As the name suggests, they are free to move about within the model executing tasks that have been assigned to them. They may transport flowitems or be used as a resource for fixed resource. Examples of task executers which move flowitems are a fork truck, crane, ambulance, wheelchair, or person. Task executers that move to fixed resources and perform other tasks include an operator required for some setup, processing, or maintenance operation.

Data, used in simulation logic, can be placed on all *FlexSim* classes.

All of the types of fixed resources, task executers, and other objects are shown in the library grid in Figure 6.5. The grid is located on the left side of the main *FlexSim* interface.

Exercises

Simulation is an applied technology and has little meaning when used to simply create models without an objective in mind. The belief that if you just build a simulation something good will happen doesn't hold true in practice. Chapter 4 of this book emphasized the need to establish a proper scope and detail level for simulation projects and discussed a methodology.

Lessons learned by actually solving problems using simulation are more likely to be remembered. The exercises in this and later chapters follow this practice. In each case the problem background and associated data are provided. The steps to follow should include an analysis of the system, a definition of how best to simulate the system using an OFD for planning, and an understanding of what simulation functions are required to resolve the problem.

Exercise 6-1 Johnson Pharmaceutical

Background

As the largest supplier of over-the-counter medications, Johnson is redesigning its cold remedy line for higher capacity in time for the upcoming flu season. The plan is for the packing line to be completely re-designed. The new plan, conceived by the corporate packaging group, is based on the "greedy" concept. In that design, boxes of individual doses are taken off the main conveyor line by the first test/wrap machine.

A second test/wrap machine takes off boxes further down the conveyor. The boxes arrive in the packing area at an average rate of 60/min. The wrap machines are rated at an average of 40/min. Based on average rates, the first two machines can handle 133% of the incoming boxes rate; however, a third will also be added as a backup. The plant engineering manager has asked you to validate the design before the plant commits to the proposed production rate.

Figure 6.19 Pharmaceutical packaging

Problem statement

Validate the new packaging line design.

Operating data

Time between arrivals of medical boxes: exponentially distributed (0,1,1) with mean of 1 second. The first parameter is the location parameter and is usually 0; the second parameter is the mean; the third parameter identifies the random number stream to use (more on setting this value later).

- Test/Wrap machine cycle times: exponentially distributed (0,1.5,2) with a mean of 1.5 sec.
- Conveyor speed: 1 unit/sec.

Expected results

- Draw an OFD for the system
- Comment on the design by running a simulation for an 8-hour shift (28,800 sec). What would you recommend to the plant manager? What rate should the plant commit to?

Modeling and analysis issues

- How should the conveyors be shown? What is the flow logic for them?
- Based just on the average rates what would you expect the capacity of the line to be and why?

Simulation results and comments are included in the Appendix.

Exercise 6-2 Lucky Air

Background

Lucky Air, a start-up airline, is committed to providing a shuttle service between the Orange County airport and the Las Vegas airport. With their fleet of regional jets, the company feels that the time is right to bring casino patrons for quick daily visits to Las Vegas. Their promise is to fly as long as there are people who want to travel. If a scheduled plane is full, another will be brought

Figure 6.20 Lucky Airlines

out; their motto is Always a Winner. They expect an increase in business as people try to find money from the slot machines during the downturn in the economy.

The owner wants to set up operations as quickly as possible so he decides to operate their check-in counter with three ticket agents: one for passengers with e-tickets, another for passengers with paper tickets, and a third for passengers purchasing a ticket. As the only engineer in the new airline, you don't think the level of service will be good and the agents will be working inefficiently. Since you don't want to verbally confront the owner, you decide on showing what might happen with a simulation.

Problem statement

You believe the proposed operation will be inefficient and you want to illustrate the system's behavior to the owner.

Operating data

Estimates of passenger demand, in terms of times between arrivals to the ticketing area and time for an agent to service each passenger type are provided in the table below. Assume the time between arrivals is exponentially distributed while agent service time is normally distributed—all times are in minutes.

Passenger type	Time between arrivals	Service time
E-ticket	mean: 5 min	mean: 3 min std. dev. 1 min
Paper ticket	mean: 10 min	mean: 8 min std. dev.: 3 min
Purchase	mean: 15 min	mean: 12 min std. dev.: 3 min

Expected results

- Fill out Parts I and II of the project template.
- Provide the following metrics based on a simulation run of 168 hours:
    - The average and maximum length of the lines for each agent's station
    - The average wait time for each type of customer
    - The average number of customers each station services per hour (throughput)

- The average utilization for each agent

- Prepare an executive summary based on conclusions drawn from the simulation.

Modeling and analysis issues

- How could the following objects be used to represent the waiting line?
  - a queue—what options would make it look like a line?
  - a conveyor
  - a flow node
- Could a processor be used as a conveyor? How? Under what assumptions?
- What are the differences in the four constructs identified above for getting passengers to the agents? Does the choice significantly impact the results?

Simulation results and comments are included in the Appendix.

Review questions

1. Identify three "nuggets"—the things you found to be the most interesting or most important—in the chapter.
2. Discuss the components included in the modeling environment of most simulation software applications.
3. Identify and define the basic elements that comprise a simulation model developed in *FlexSim*.
4. Describe how the movements of flowitems between fixed resources occur within a *FlexSim* model.
5. Describe the difference between fixed resources and task executers (the two basic categories of resources used in *FlexSim*).
6. Discuss how the *FlexSim* object structure enhances model development.
7. Select several *FlexSim* objects; for each one, identify the various states that the object can be in during a simulation.
8. Compare the behavior of accumulating and non-accumulating nveyors.

9. Construct a *FlexSim* model of the Compu-Help system described in the Chapter 5; however, assume the devices do not crash. Run the model for 10 hours and determine the following:
    a. Average number of calls waiting
    b. Average time a caller waits before being helped
    c. Percentage of customers lost due to the system's limitation of only being able to keep six calls waiting.
10. Modify the basic Compu-Help model so that the maximum queue capacity is 1000. Run the model for 5 hours, stop it (do not reset), and note the average wait time. Run the model for 5 more hours, stop it (again, do not reset), and note the average wait time. Repeat this eight more times. Plot the average wait time versus cumulative run time. What can you infer from the graph?
11. Discuss how a model's behavior differs when a queue, conveyor, and flow node are used to move flowitems between two fixed resources.

Index

E

F

G

H

I

J

L

M

N

O

P

U

V

W