Applied Simulation

Modeling and Analysis using FlexSim

Fifth Edition

Compatible with FlexSim 2017 LTS

Malcolm Beaverstock, PhD

Allen Greenwood, PhD, PE

William Nordgren, MS CIM

FlexSim is chosen by the authors because of its comprehensive functionality and its ease of use. It is 3D simulation software that facilitates modeling, analysis, and visualization of systems in manufacturing, material handling, healthcare, warehousing, mining, transportation, logistics, etc. It effectively provides the means to: (1) address the physical and logical aspects inherent in simulation modeling and (2) experiment with, and analyze, simulation models. *FlexSim* closely aligns with the purpose of this text—namely to understand simulation technology and to use it to solve problems.

Exercises in the book require either an academic license of *FlexSim* or a commercial license of *FlexSim*. However, the free trial version of *FlexSim* (*FlexSim Express*) can be used for exercises in Chapters 3 and 4. Special files for the book are available in the [BookDownLoadFiles.zip] file on the FlexSim Education/Student website. The first two chapters and the last two chapters of this textbook do not require any software, including *FlexSim*. This book is not intended as a software manual; therefore, refer to the *FlexSim* Help files for more details on software operation.

.

www.flexsim.com

Canyon Park Technology Center

Building A – Suite 2300

Orem, Utah 84097 USA

Phone (801) 224-6914 • Fax (801) 224-6984

sales@flexsim.com

Applied Simulation: Modeling and Analysis using FlexSim

by Malcolm Beaverstock, PhD
Allen Greenwood, PhD, PE
William Nordgren, MS CIM

Published by FlexSim Software Products, Inc., Canyon Park Technology Center, Building A Suite 2300, Orem, UT 84097 USA.

FlexSim software and books may be purchased for educational, business, or sales promotional use. For more information, please contact our sales department: +1-801-224-6914 or *sales@flexsim.com*.

Editing: Katherine Bobo, Markus Cueva

Managing Editor: Markus Cueva

Cover Design: Stacy Geisberger, Kris Geisberger, and Ben Wilson, with special thanks to Olivier Pellegrin, Brenton King, TALUMIS, and others who provided screen captures of actual *FlexSim* simulation models.

Printing History:

January 2011:	First Edition, version 1 (pre-release)
August 2011:	First Edition, version 2 (pre-release)
September 2011:	First Edition, version 3
January 2012:	Second Edition
July 2012:	Third Edition
October 2014:	Fourth Edition
June 2017:	Fifth Edition

ISBN: 978-0-9832319-5-0

Preface

Simulation is an applied technology that is especially useful for analyzing and solving problems. Applying simulation begins by being clear on the problem definition, the reasons for simulating, and the expected outcomes. Simulation with no objective is counterproductive.

A person using simulation then must balance their understanding of the problem with their knowledge of the details of simulation: the underlying simulation concepts, application software, and the analysis methodologies that are employed. The most effective way to learn how to successfully use simulation is through learning how to apply it.

A major challenge in teaching applied simulation is the question of how to effectively blend and balance an understanding of fundamental principles and concepts with the practical side of building simulations. The intent of this book is to help bridge that gap and improve the effectiveness of simulation courses. Not all readers of this book will become simulation experts, but hopefully they will want to utilize the technology to help them or others make better decisions. Consequently the material takes the reader through three levels of users: Occasional, Intermediate, and Advanced.

Modern simulation applications allow the user to accurately represent an environment both as a visualization of the physical elements as well as the underlying, often complex, logical structure. Using both aspects of a simulation greatly increases its effectiveness in gaining acceptance and solving problems. *FlexSim* was chosen for use with this text because of its ease-of-use and rich visual and logical functionality that allows users to focus on simulation concepts and methods. This is not intended to be a *FlexSim* manual as the *FlexSim* Help files and tutorials are more than adequate for that purpose.

The first three chapters provide a background for the practice and use of simulation and provides a base for all user levels. Chapters 4 through 6 focus on the needs of an Occasional User, who may not want to be a simulation professional but wants to try using simulation. These chapters focus on the basics of simulation software, statistics, and model development. They show the reader how to use the standard functionality of the software to build simple models that can incorporate a realistic level of complex logic.

Chapters 7 thru 11 introduces the Intermediate user to the additional theory and background needed to address the very practical issues of reliability, randomness, and output analysis that are critical to a successful simulation.

The Advanced user will find in Chapters 12 through 15 concepts and examples that go beyond the standard software functionality, with a focus on how to easily add robustness to model logic. This section also adds the ability to integrate aspects that exhibit fluid flow characteristics into the simulation.

The final section, Chapters 16 and 17, addresses professional practice - topics that focus on successfully implementing a simulation project. This guide is usually not covered in other texts but is critical for successfully carrying out simulations in a corporate environment. This section can be covered anytime in a course, preferably just prior to starting a course project.

Contents

Preface . . . . . **VII**

Chapter 1: Simulation as a Tool for Understanding **13**

Section 1-1 What is simulation? 13
Section 1-2 The problem with waiting time. 15
Section 1-3 Working in a global, dynamic environment 24
Section 1-4 Simulation in everyday life 29
Section 1-5 Where is the money? 33
Section 1-6 When and where to use simulation 36
Section 1-7 Simulation users 42
Section 1-8 Summary 47

Chapter 2: Introduction to Modeling and Analysis of Queueing Systems **48**

Section 2-1 Prevalence of Queueing 50
Section 2-2 Basic Elements of Queueing 51
Section 2-3 Characteristics of Queueing System Elements. 53
Section 2-4 Measures of Queueing System Performance 60
Section 2-5 Means to Study of Queueing Systems 62
Section 2-6 Analytic Models of Queueing Systems 63
Section 2-7 Analyzing the Behavior of Queueing Systems 67
Section 2-8 Simulating Queueing Systems 74
Section 2-9 Summary 84

Chapter 3: Using Simulation to Solve Problems **93**

Section 3-1 Using a simulation 93
Section 3-2 Simulations with random events 94
Section 3-3 Examples and exercises. 96
Exercise 3-1 Coasting around 98
Exercise 3-2 Martian Transfer Station 100
Exercise 3-3 Slime Inc. 102
Exercise 3-4 Danson Electronics. 104
Exercise 3-5 Maritime Ltd.. 106
Exercise 3-6 Zombie Batteries, Inc.. 108

Chapter 4: Building Basic Simulation Models **111**

Section 4-1 Essential features of discrete-event simulation models. 112
Section 4-2 Simulation environment. 114
Section 4-3 Simulation components 117

Section 4-4 Fixed resources 119
Section 4-5 Transporting items 120
Section 4-6 Editing objects 122
Section 4-7 Making connections 125
Section 4-8 Moving flowitems. 127
Section 4-9 Creating simple learning models 132
Section 4-10 Single-run statistics 133
Exercise 4-1 Johnson Pharmaceutical 136
Exercise 4-2 Lucky Air 138

Chapter 5: Adding Model Logic and Managing Data 141
Section 5-1 Assigning attributes to objects 141
Section 5-2 Adding logic. 142
Section 5-3 Managing data through tables 146
Section 5-4 Grouping and ungrouping flowitems 147
Section 5-5 Reusing custom objects 150
Exercise 5-1 More Lucky Air 152
Exercise 5-2 Even More Lucky Air. 154
Exercise 5-3 Hampton International. 156

Chapter 6: Improving System Representation 159
Section 6-1 Mobile resource objects (task executers). 159
Section 6-2 Using Lists for decision making 165
Section 6-3 Displaying information on the screen. 171
Exercise 6-1 Steve's Stone Cutting 176
Exercise 6-2 Treasures-4-U 178

Chapter 7: Incorporating Resource Availability and Reliability. 181
Section 7-1 Establishing Time Tables 182
Section 7-2 Performance 184
Section 7-3 Reliability 185
Section 7-4 Estimating MTBF and MTTR 187
Section 7-5 Simulating machine failures 188
Section 7-6 Setting MTBF and MTTR in a simulation. 190
Section 7-7 Using personnel for repairs. 192
Section 7-8 Surge 193
Exercise 7-1 Keggler's Brew 194
Exercise 7-2 Chairs for Tots 197

Chapter 8: Modeling Randomness 203
Section 8-1 Importance of probability distributions in simulation 204

Section 8-2 Selecting probability distributions based on sample data . . . 206
Section 8-3 Selecting probability distributions in the absence of sample data. 216
Section 8-4 Sampling from Continuous Probability Distributions 218
Section 8-5 Sampling from Discrete Probability Distributions 222
Section 8-6 Generating Random Numbers 223
Section 8-7 Putting It All Together— Obtaining Random Samples from Probability Distributions 226

Chapter 9: Simulation Mechanics— How Monte Carlo and Discrete-Event Simulations Work. 235
Section 9-1 Monte Carlo simulation 236
Section 9-2 Discrete-event simulation. 243

Chapter 10: Fundamentals of Output Analysis 265
Section 10-1 Statistical inference from simulation models 266
Section 10-2 Key tactical experimental design decisions 273
Section 10-3 Creating experiments using *FlexSim's* Experimenter 285

Chapter 11: Multi-scenario Experimentation and Optimization . . . 295
Section 11-1 Definition of case study and preliminary analysis. 296
Section 11-2 Comparison of two alternatives 302
Section 11-3 Reducing variability in the samples using Common Random Numbers (CRN). 306
Section 11-4 Comparison of multiple alternatives using Bonferroni's method 310
Section 11-5 Introduction to Optimization. 314

Chapter 12: Logical Relationships Among Objects 333
Section 12-1 Hierarchical software architecture 334
Section 12-2 Understanding how objects work 337
Section 12-3 Scripting in *FlexSim* 339
Section 12-3 Scripting tips 345
Exercise 12-1 Hilltop Steel Works 358
Exercise 12-2 The Crafty Framer 360

Chapter 13: Customizing Models 363
Section 13-1 Communicating between objects 363
Section 13-2 Label Tables 368
Section 13-3 Customizing object placements 370
Exercise 13-1 Fister's Express 374
Exercise 13-2 Peoples Surgery Center. 376

Chapter 14: Advanced Logic Functionality and Representation 381
Section 14-1 Basic concepts of Process Flow elements 382
Section 14-2 Defining characteristics of Process Flow elements 383
Section 14-3 Building a Process Flow 383
Section 14-4 Creating zones with Process Flow elements 388
Section 14-5 Synchronizing Process Flow elements with the 3D layer . . 391
Exercise 14-1 Greyson Operations 398
Chapter 15: Simulating Fluid Flow 403
Section 15-1 Basics of fluid-flow simulation 403
Section 15-2 Fluid objects 405
Section 15-3 Specifying Fluid Operations 407
Exercise 15-1 James Peanuts 412
Exercise 15-2 Western Grain 415
Chapter 16: Applied Simulation Environment 419
Section 16-1 Confidence 419
Section 16-2 The Simulation Modeling and Analysis (SMA) Life Cycle 421
Section 16-3 The modeling and analysis process 422
Section 16-4 Roles in a SMA project 425
Section 16-5 Model-based decision-support systems 428
Section 16-6 Simulation and other tools 429
Section 16-7 The simulation software marketplace 432
Section 16-8 Simulation modeling and analysis success factors 436
Chapter 17: Managing a Simulation Project 441
Section 17-1 The starting point 441
Section 17-2 Define the system and the reason to simulate 443
Section 17-3 Prepare a conceptual model 448
Section 17-4 Initial sizing of resources 464
Section 17-5 Building the simulation 465
Section 17-6 Results and analysis 466
Section 17-7 Example of a project template 469
About the Authors 483
Index 486

Chapter 3 Using Simulation to Solve Problems

Objective

Experience how simulation is used by interacting with pre-built simulations, analyzing results, and recommending solutions to specific operational questions.

It is important for all stakeholders (i.e., those who own the problem being simulated, those supporting the simulation, those building the simulation, and those who will make decisions based on the simulation) to understand the power of simulation and how it can be used to solve problems. This chapter uses six diverse models to illustrate how simulation can be applied to improve system performance. All models employ a custom interface so that the users play the role of decision maker and Occasional User. The focus is on analysis and decision making and not on software and model building—that comes later in the book.

Section 3-1 Using a simulation

Simulation is an applied technology. There is little value to a simulation built without a reason or a problem to be solved; therefore, simulation must be experienced to be appreciated and understood. This chapter is an introduction to simulation applications through the eyes of an Occasional User.

An Occasional User, as defined by this book, is a user who doesn't use simulation on a regular basis but appreciates the value of simulation. Usually this person's main responsibility is something other than simulation—a manager, lead engineer, team leader, etc. The Occasional User is not proficient in building simulation models but, with a little review, can make use of a simulation built by others, especially if the simulation is intuitive. It is assumed, however, that the user is familiar with some of the basic concepts of simulation.

The Occasional User knows enough about simulation to create a functional specification for others based on his or her knowledge of the system being simulated. As the "owner" of

the problem or issue to be studied by the simulation, the Occasional User can detail the simulation requirements, define the scope, and establish the metrics for the analysis.

This chapter demonstrates how simulation can be used by the Occasional User. Most simulators on the market today are set up for Occasional Users to run simulations and even provide "run-time" versions of the software. While *FlexSim* software is used in this book, the methods for doing the exercises in this section should be typical of other products as well.

The simulations used in the following examples represent models built by Intermediate and Advanced Users. They are usually prepared for Occasional Users to use and therefore can be opened and run without any knowledge of the underlying software application. Details of the application environment for *FlexSim* are contained in the help files.

Section 3-2 Simulations with random events

The challenge for the Occasional User is to understand the simulation he or she is viewing and to interpret the results. The simulation interface will normally allow the user to change certain variables and view previously defined metrics. For a simulation that contains random events, this can be confusing.

Almost every event in the world will happen differently every time it occurs. Each event is unique and influenced by its environment. This phenomenon is characterized in simulation models by using random events defined by probability distributions. For example, distributions are used to represent arrival processes, such as patients coming to an emergency room; process times, such as the time to assemble products; transportation times, such as the time to move material from one location to another; interruptions, such as machine breakdowns, etc. These statistical distributions can be developed based on historical data or by professional estimate.

As shown in Figure 3.1, and as described above, input to a simulation model involves random variables; therefore, the model is considered stochastic since it uses random variables. These stochastic inputs are a subset of the uncontrollable variables in the model (i.e., they are not under the control of the decision maker).

Controllable factors, or decision variables, are those that are manipulated in the model to improve a system's performance (e.g., number of workstations, number of transporters to move products, processing rules). Since the inputs to a model are stochastic, the output or performance measures are also stochastic. Care must be taken to evaluate stochastic output. One run of a model represents just one way a system may behave; therefore, a decision should not be made on whether the model is valid or if one alternative is better than another based on a single run since one run could represent an extreme case.

The bottom line is that a simulation that utilizes random events must be either run for a long period of time or many times (i.e., for many repetitions or replications). This is the only way to obtain data that can be used to make an accurate decision. Statistical methods can be used to determine the length of a simulation run or the number of times a simulation

model needs to be run to ensure a specific level of confidence in the estimate obtained from the simulation model.

Figure 3.1 illustrates how simulation models are used. Once the values of the decision variables are set, the model is run and output is obtained. The user then changes the decision variables (based on the output from the initial run as well as the specified system performance criteria) and re-runs the model. The user continues this iterative process until a satisfactory system configuration is realized.

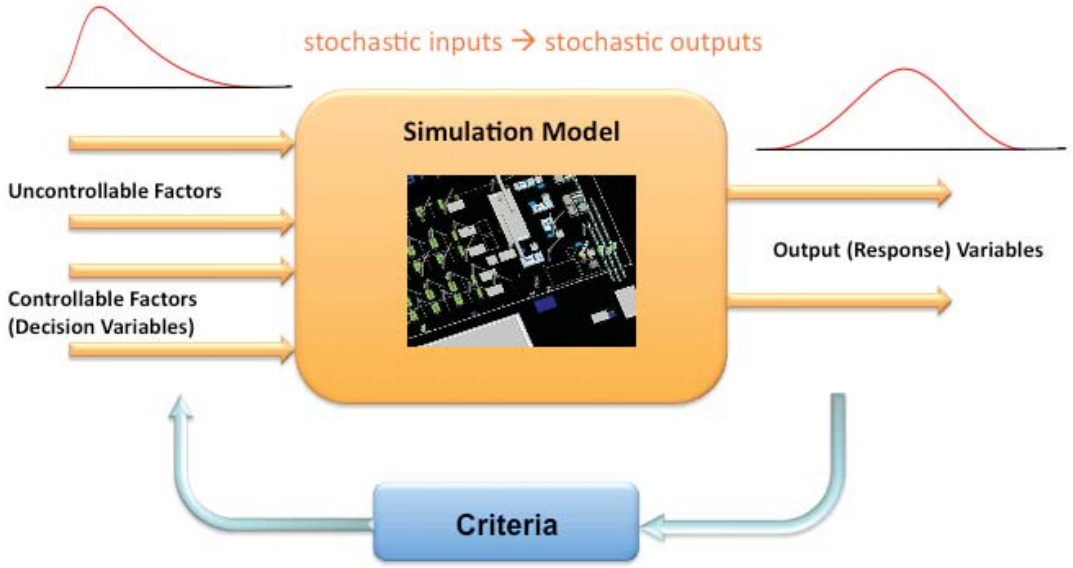

Figure 3.1: Stochastic simulation.

Almost all the simulation models in this book include some degree of randomness. An option to use the same random number streams was selected for each example so that variable changes could be more easily compared; however, to maintain focus on the use of simulation by the Occasion User, only a single or a few runs will be requested to answer the questions. It needs to be stressed that this is not meant to downplay the importance of the statistical analysis, which is a critical part of any simulation project; however, it is common for Occasional Users to simply run the simulation and collect data without regard to the underlying statistics. This is because he or she typically uses a model developed by someone else, and the experimental conditions are pre-set within the model execution environment. Simulation models are commonly embedded in decision support systems that free the user from connecting to required data sets, formatting results, and setting up experiments.

Three chapters in this book deal with statistical analysis—one for input data analysis (Chapter 8) and two for analyzing simulation output (Chapters 10 and 11). Chapter 11 includes a discussion on the use of the Experimenter functionality included in *FlexSim*. The Experimenter facilitates carrying out multiple simulation runs and performing statistical analyses.

Chapter 6 Improving System Representation

This chapter extends modeling capability in several key areas. Mobile resources are one of the main classes of objects in any simulation software. In *FlexSim*, mobile resources are referred to as task executers. These are especially versatile and powerful objects. The first section of the chapter describes the various types of task executers, including their basic operation and the main properties that drive their behavior.

Objective

To utilize additional functionality available in simulation applications that enhance modeling and enable help others to better visualize, understand, and have confidence in the simulation results.

Often simulations may focus on simulating activities that have no direct visualization. While standard objects contain a wide variety of logical choices in the drop down menus, trying to make use of them can be difficult. *FlexSim*, in their 2016 version of the software, has added new capabilities to handle logical expressions without compromising the value of good visualizations. In this chapter, the use of Lists is introduced. Additional use of the advanced logic capability is discussed in Chapter 14.

In order to enhance visualization, additional information oftentimes needs to be displayed on the simulation surface. This information may include statistics, state status, descriptions, etc. Sections 6-3 and 6-4 describe how this information can be displayed in *FlexSim*.

Section 6-1 Mobile resource objects (task executers)

In addition to the objects previously described to move flowitems, simulations often require mobile resources to move or carry items. These resources can include forklifts, automated guided vehicles (AGVs), or operators. Simulations may also require mobile resources to carry out certain tasks, such as setting up or repairing a piece of equipment. The mobile resource objects, also referred to as task executers in *FlexSim*, can perform various functions:

- Physically move over the simulation surface

- Carry or transport flowitems between objects
- Execute a series of tasks
- Be assigned to an object in association with a particular operation (e.g., process setup, processing, maintenance)
- Distribute tasks to other task executers
- Follow a path created by network nodes

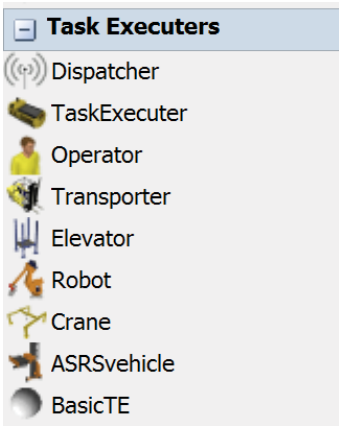

Figure 6.1: Task executers.

As shown in Figure 6.1, *FlexSim* offers a variety of types of task executers in the standard Discrete Objects library. In terms of their basic functionality, the objects are very similar. Note that there is a task executer object within the general class of task executers.

The task executer object and operator objects are very similar. The main difference is that the operator can be animated to represent walking and carrying an item. The transporter object, analogous to a fork truck, can raise flowitems vertically using its forks, as well as travel the surface. An elevator object only moves flowitems vertically, or in the Z direction. A robot moves flowitems between two points on the model surface from a fixed position using its arm (the base of the robot object does not actually move). Cranes and ASRS vehicles are more complex task executers. Most beginners use the basic task executers such as operators and transporters; however, it is helpful to be aware of what is available for more complex modeling projects. All task executers can travel along the model in three dimensions if attached to a path network.

As their name suggests, the primary function of task executer objects is to execute task sequences. A task sequence contains a series of specified tasks that are executed in sequence, has a priority value that indicates its importance to other task sequences, and can preempt or halt a task sequence that is being performed. Task sequences can be created automatically (e.g., when a fixed resource "uses transport"). In the case of Use Transport, the following tasks are executed:

1. Travel to the object that contains the item to be moved
2. Load the item from the calling fixed object to the task executer
3. Break; that is, check to see if there is some other task waiting (e.g., load another available item if the task executer has capacity)
4. Travel to the destination object
5. Unload the item from the task executer to the receiving fixed object

Another automatically generated task sequence occurs when assistance is needed to perform a process. In that case the task sequence is

1. Travel to the object that needs assistance with a process;
2. Utilize the task executer for the duration of the process.

One powerful feature of *FlexSim* is that the modeler can construct his or her own custom task sequences. This is an advanced topic; for more information, refer to the *FlexSim* User Manual. Task executer objects execute a task sequence if there is currently no active task sequence or if the incoming task sequence has a higher priority. Otherwise, the task sequence is dispatched to an associated task executer or it is queued for execution until the current task sequence is completed.

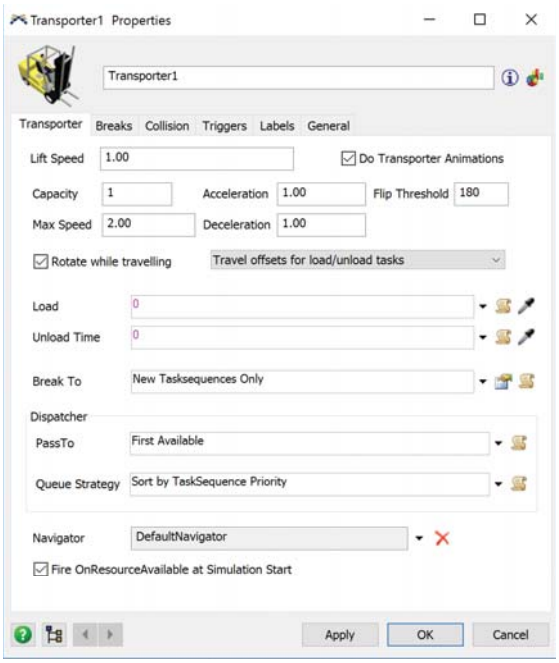

Figure 6.2: Task executer's parameters.

As shown in Figure 6.2, the main interface window for task executers is very different from other objects. The one shown is for a transporter type of task executer. All task executer objects have two main sections, one pertaining to operating properties of the object and the other addressing task executer behavior.

The first section of the interface is used to set values concerning the operation of the task executer; the variables vary somewhat depending on the role that the task executer performs.

Task executers that are being used to convey flowitems have the following built-in attributes:

- Capacity: This indicates the maximum number of flowitems that can be carried or transported at a time.
- Load/Unload time: This attribute gives the time, in simulation units, to load a flowitem at the originating object and the time to unload it at the destination object.
- Lift Speed: This indicates the speed in grid units per time unit.
- Travel offsets for load/unload tasks: If this option is not selected, the task executer will go to the origin of each object to load or unload flowitems. If selected, the task executer will go to the origin first and then move to the exact point where the flowitem is to be loaded or unloaded.
- Flip Threshold: When the angle between the transporter and the destination meets or exceeds this value, the transporter will "flip" to a mirror image in order to be facing the correct direction. This is for advanced visualization effects only.
- Attributes when being used for any task are as follows:
- Rotate while Traveling: If checked, the transporter will rotate as needed in order to orient itself in the direction of travel. This is for visualization effects only.
- Speeds (Max speed, Acceleration, Deceleration): The travel speed is in grid units per time unit and grid units per time unit squared, respectively
    - The acceleration controls how fast a task executer reaches its maximum speed or how fast it slows down when it reaches its destination. Unless a very precise

Exercise 7-2 Chairs for Tots

Background

Chairs for Tots builds children's chairs that are sold through major outlets and over the internet. The production facility is just keeping up with their order rate but is also trying to keep operating costs down. Work in process at the end of the week is continued the next week. Ideally, operators help clean the area at the end of the week; however, if there are orders still to be completed, the cleaning operation becomes difficult. The plant doesn't want to add overtime but management feels they have to do something to increase the weekly output and still have time to clean and service the equipment.

Problem statement

What will be the impact of higher order rates on the operations?

Operating data

The base order rate for chairs is approximately one every 25 minutes (distributed exponentially) but is expected to increase to one every 20 or even 15 minutes during a peak period. The plant starts operations at 8 a.m. Monday morning and runs 24 hours a day. It stops taking orders at noon on Friday so that the work in progress can be completed, the area cleaned, and the workers sent home at 4 p.m.

There are 4 major production steps:

1. For each order, a set of raw materials is conveyed to the cutting machine loading queue. An operator moves the wood from the queue to the cutting machine, which operates automatically.
2. Once cut, the cutting operator places the set of parts on a conveyor. The parts move to another queue where they wait to be assembled and painted. The assembly operator takes the parts and then assembles and paints each chair.
3. The painted chairs are conveyed to an oven to dry and set the glaze. The oven can hold up to five orders. Therefore, five orders are usually accumulated and placed in the oven at one time; however, any order shouldn't wait more than 150 minutes to be placed in the oven.
4. When dry, the orders are transported by conveyor to the packing area where they are inspected and prepared for shipment.

Operating times (in minutes; the varying times reflect the size distribution of the orders) can be found in Table 7.4.

Operation	Attribute – Minutes	Distribution
Order arrival rate	Mean = 30	Exponential
Cutting	Min 10; Max 20	Uniform
Assembly	Min 10; Max 20	Uniform
Heat Treatment	60	Constant
Inspection	Mean 20; Std 5	Normal

Table 7.4: Operating times.

Downtime occurs in a number of ways. The cutter and assembly operation both have frequent and infrequent downtimes. The frequent downtimes can be repaired by the operator. The operator will always stop the present job to repair the machine. The infrequent downtimes (changing a cutter blade and repairing the paint sprayer) have to be handled by the plant mechanic, who responds to breakdowns as they occur. The plant mechanic also has to fix problems in the other plant areas.

Operation	Time between failures (minutes)	Distribution	Time to repair (minutes)	Distribution	Repair Person
Cutter	Min. 20; Max. 80	Uniform	Min. 1; Max. 4	Uniform	Cutter Operator
Assembly	Min. 35; Max. 55	Uniform	Min. 1.2; Max. 3.0	Uniform	Assembly Operator
Oven	Mean 420; Std. Dev. 20	Normal	Mean 5; Std. Dev. 2	Normal	Mechanic
Cutter blade change	760	Constant	Min. 10; Max. 15	Uniform	Mechanic
Paint sprayer	Min. 400; Max. 560	Uniform	5	Constant	Mechanic
Other plant areas	Mean 100	Exponential	Min. 20; Std. Dev. 5	Normal	Mechanic

Table 7.5: Historical records for downtimes (minutes).

Material travel times

- Order entry to cutter: 3 minutes
- Cutter to assembly: 3 minutes
- Assembly to oven: 3 minutes
- Oven to packing: 5 minutes

Capacities

- Conveyor to cutter: 10
- Queue before cutter: 10

- Conveyor to assembly: 10
- Queue before assembly: 5
- Conveyor to oven: 10
- Conveyor to packing: 10

Expected results

- Run the simulation for 10 workweeks. Although in practice any incomplete orders would carry over to the next week, consider each week a separate run starting with a clean line.
- Report on production levels as well as how many orders are still in the system at the end of the week. The only operating personnel who need to be considered are the cutter operator, assembly operator, and plant mechanic.

Modeling and analysis issues

- What level of detail is needed? Should all chairs in an order and their parts be tracked?
- What should be the basic time unit?
- How can you start and stop the simulation?
- Although the combiner object can perform batching, is it a good choice here?
  - What other object can handle batching?
  - How can the max waiting time be handled?
- What is the best way to validate that the components are working correctly before adding the complexity of the downtime?
- How can the "rest of the plant" be simulated and given a downtime for the mechanic to fix?
  - Which downtimes will pre-empt other jobs?
  - How is the person to service the downtime selected?
- What performance measures are important?
- What can be done to improve operations so that all jobs are finished by 4 p.m. on Friday?
  - The union wants an additional mechanic added.
  - The cutter operator wants an automatic infeed from the cutter queue.
  - The lean team suggests changing the way orders are entered into the oven-even suggesting converting to a five-lane continuous oven.
- What are the issues with higher order rates and what can be done to improve operations?

Chapter 7 Review Questions

1. Identify three "nuggets"—the things you found to be the most interesting or most important in the chapter.
2. Name two other forms of planned down time that can be simulated with a time table.
3. Describe three different performance measures, such as those listed in Section 7-2, including one that is not based on a rate measure, and give an example of each as applied to an operations system.
4. Discuss how does reliability can impact system performance.
5. Discuss the difference between "availability" and "reliability."
6. When including reliability in a simulation model, it is important to understand the definition of the word time in mean time between failures. Discuss why this is important in a simulation model.
7. Discuss the different times that can be referred to in mean time to repair.
8. Describe the different ways you can estimate MTBF and MTTR.
9. Explain what competing failures are and how they can be simulated.
10. Assume the reliability of a machine is given by

    $$R(t) = e^{-0.01t},\ where\ \lambda = 0.01$$

    that is, the failure rate is 1 per 100 hours.

    a. Describe in words the meaning of R(t).
    b. What is the probability that the machine will run 200 hours before failing?
    c. For this case does the answer change whether the previous failure occurred 10 hours ago or 300 hours ago?
11. Describe the two ways to enter reliability data in FlexSim. How would you set MTBF and MTTR for three machines that share the same distributions?
12. What does the appearance of a gold box around a piece of equipment signify?

13. Describe what happens when the Preempt box is checked for an operator who is being used to repair more than one piece of equipment.
14. Explain the basic concepts of using a surge to protect from down time.

Chapter 10 Fundamentals of Output Analysis

Objective

To enable users of simulation to correctly analyze output and interpret results that are obtained from simulation models, especially considering the significant variation that is inherent in those models.

A simulation model is used to understand the behavior of a system in order to support decision making and solve problems. Simulation models are especially used to understand behavior when changes to the system are considered. In that sense, a model is like a laboratory, in that it is used for experimentation. Experiments are conducted by changing inputs to a model, oftentimes referred to as decision variables, and observing and analyzing the effect of the input settings on system behavior. The effects are characterized by performance measures and are the results, or output, of a simulation model.

Analyzing output from simulation models must be done with care because of the complexity of the models and the systems they represent. This complexity is due to the interactions among system/model elements, inherent variabilities and uncertainties, and dynamics. In this book, the analysis of simulation model output is discussed in two basic parts: (1) fundamentals of output analysis and the analysis of a single scenario and (2) analysis of multiple scenarios. A scenario is a set of specific value settings for the input variables that are being considered; the value settings result in output from the simulation model that represents measures of system performance.

The first part of simulation output analysis is discussed in this chapter, and the second part is discussed in the next chapter. This chapter begins with a brief discussion of how output analysis fits into the overall simulation modeling and analysis process. Subsequently, this chapter

- describes the analysis of output from simulation models in the context of basic statistical inference;

- discusses key tactical experimental design decisions that must be made to effectively use simulation models for analysis;
- introduces FlexSim's Experimenter, a very powerful tool for analyzing model output and conducting experiments with simulation models.

While analysis begins once a simulation model has been developed, validated, and verified, planning the analysis must start at the beginning of the simulation process. As defined and discussed in Chapter 16, simulation modeling and analysis (SMA) is a process. A portion of that process is shown in Figure 10.1. The SMA process starts with a clear set of objectives for why the simulation project is being conducted. From these objectives, the questions that need to be answered via simulation should become evident. Typically the objectives identify the system performance measures that will be used as a basis for decisions and choosing among alternatives. These measures are the output of the simulation.

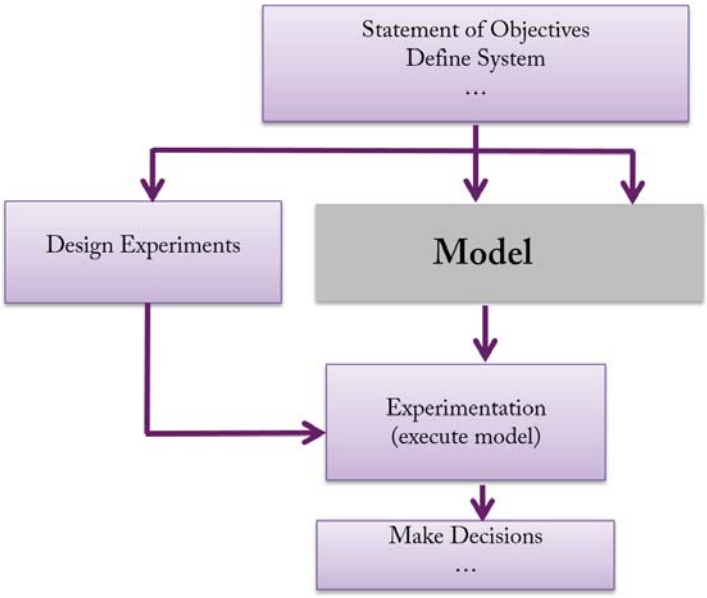

Figure 10.1: A portion of the Simulation Modeling and Analysis process.

Also as shown in Figure 10.1, the design of the experiments begins in parallel with the development of the model(s). The design process includes both the definition of (1) key performance measures, output of the simulation, and (2) the variables that will be changed across the various alternatives under consideration, the input to the simulation.

Section 10-1 Statistical inference from simulation models

As has been discussed throughout this book, many system characteristics (process times, reliability of equipment, arrival pattern of customers, etc.) are represented as probability distributions in simulation models. As such, simulations are driven by the use of random samples from probability distributions, along with logic and physical characteristics, to represent a system's operation and behavior; Since the systems are stochastic or probabilistic, so are models of those systems. As shown in Figure 10.2, stochastic inputs lead to stochastic outputs; that is, since input to simulation models are random variables, even if there is only one, then the output—response variables or performance measures—of the system are also random variables. Output from simulation models therefore must be analyzed using basic statistics; in other words, inference must be made about the true characteristic of a system based on a sample of information from that system. Random sampling from probability

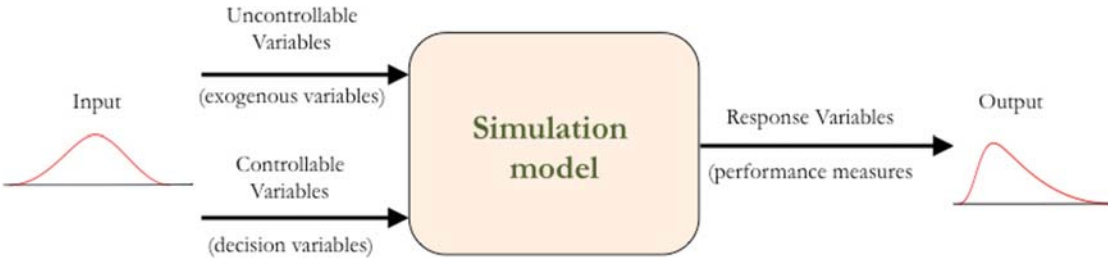

Figure 10.2: Stochastic input lead to stochastic output.

distributions was discussed in Chapter 8. Recall, probability distributions can be parametric (e.g., normal, exponential, Weibull) or non-parametric (e.g., empirical) and can be specified based on sample data or in the absence of such data.

Figure 10.2 also shows that there are two basic types of input to simulation models—uncontrollable variables and controllable variables. Controllable variables, also called decision variables, are those that are under immediate control of the decision maker. These are the variables that are subject to change in a simulation in order to compare alternatives. Some common examples of decision variables include the number of servers or operators to use, the queue discipline to follow (e.g., FIFO, priority), and the location of the equipment and other resources. Controllable variables are typically not random variables. On the other hand, uncontrollable variables are oftentimes random variables and are not under the immediate control of the decision maker (e.g., process times, reliability, and inter-arrival times). Of course, the distinction between controllable and uncontrollable depends on context. For example, if three alternative means for performing a process are being considered, then the means would be controllable. The means are not random variables, but the times resulting from their distribution are.

When making a decision, the decision maker would like to do so with perfect information. They would like to know the true characteristics of a system; or, in statistical terms, they would like to know the population values of these characteristics. In order to do so, all values of the characteristic need to be known, which is impossible for most systems, so the population characteristic, or parameter, needs to be estimated. This estimate is oftentimes based on statistics obtained from a sample. In general, the population is all members of a specified group, and a sample is a subset of the members.

The best way to understand the fundamental output analysis concepts is via example. We start with a simple example from Chapter 2. Consider a single-server queueing system with inter-arrival times that are exponentially distributed with a mean of 1.0 minutes and service times that are exponentially distributed with a mean of 0.9 minutes; hence, the average server utilization is 90% (0.9/1.0). The system also processes customers on a FIFO basis, there is infinite queue capacity, and the server is always available (no downtime). This is a basic M/M/1 queueing system. If the system is such that it meets the assumptions of the M/M/1 Queueing Theory model, then simulation is not needed—the performance measures are obtainable from the equations provided in Chapter 2. We use simulation here, however, to demonstrate the importance of the experimentation parameters and to compare the experimental results to known population values (i.e., those obtained from the M/M/1 formulas).

Chapter 13 Customizing Models

Most simulation languages have the capability to send information from one object to another in order to facilitate building complex logic. In *FlexSim* this communication is accomplished through messages that can be dynamically sent from any one point in the model to another. As simulations become more complex, additional ways of storing and dynamically displaying information become important. Additionally, enhancing the 3D representation of objects to better represent the actual process increases the acceptance and understanding of the simulation.

Objectives

To describe why communication between simulation elements is important; to use messaging between objects to carry out custom logic; to define additional ways of storing dynamic information while a simulation is running; to describe methods for enhancing the visual elements of a simulation.

Section 13-1 Communicating between objects

Communicating between objects with messages can be a convenient and efficient way of sending information to coordinate and control the simulation. A message can be sent from any trigger on any object in the model to any other object in the model at any time during the simulation. The OnMessage trigger (located in the Triggers tab) on each object executes when it receives a message.

There are two basic commands for sending messages from anywhere in the simulation:

```
sendmessage(to_object, from_object, param1, param2,
param3);
```

```
senddelayedmessage(to_object, delay_time, from_object,
param1, param2, param3)
```

Messages are either sent immediately or after a delay. The first command listed above sends a message when the object is executed in the simulation. The delay message command sends the message after a specified delay time.

The `sendmessage()` command sends the message immediately. This means that during the same calculation cycle, the message will be sent and the received message will be executed before the statement following the command is executed. At times such immediate response is required for synchronizing events that are very closely coupled.

There may be times, however, when the received message should be executed only after all other calculations are completed for the time cycle. In such cases, the correct command to use is `senddelayedmessage()` with a delay time of 0. A delay time of 0 will send the message at the end of the current calculation cycle of all objects. It is a good choice when an immediate message is desired but should wait until the states of all objects are updated.

Each message can contain up to three numeric parameters.

On the receiving side, the actions to be taken can be described by the choices in the drop down menu or through the script window; however, the script in the OnMessage trigger contains only a single line of code that defines the treenode variable current. To access the message parameters that may have come with the message, the command `msgparam(n)`, where n is the parameter number 1,2, or 3, must be added to the logic. This command can be used in an assignment statement, such as `double mp1 = msgparam(1);` or directly in the logic, such as `switch(msgparam(3))`.

Messaging examples

Messages expand the scope of logic and allow the simulation of events that would otherwise be difficult. They are often used in simulating events that have to be synchronized. Consider a just-in-time assembly area where products are assembled to order. As a part is being produced to meet a particular order in one manufacturing cell, a subassembly that will be added later has to be started in another cell, as shown in Figure 13.1.

In this case the first assembly unit sends a message at some point in the production cycle to the second unit building the subassembly. A variable containing the order number of the item being built is passed as a message parameter. Using the OnMessage trigger, the second unit looks up what is required and sets up its own parameters to create the required subassembly.

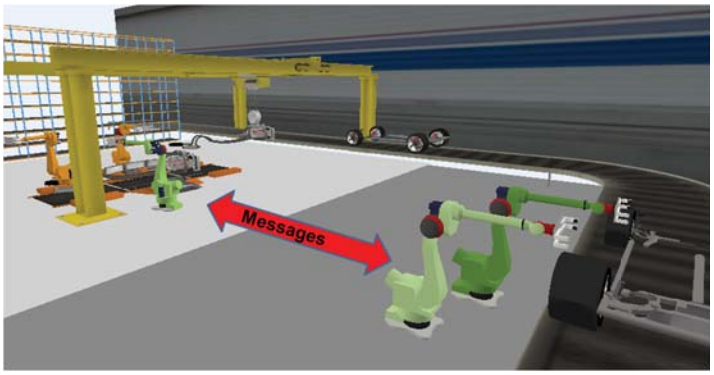

Figure 13.1: Manufacturing Cell.

Another example of the use of messages involves a machine that, when started from a stopped condition, goes through a speed ramp-up

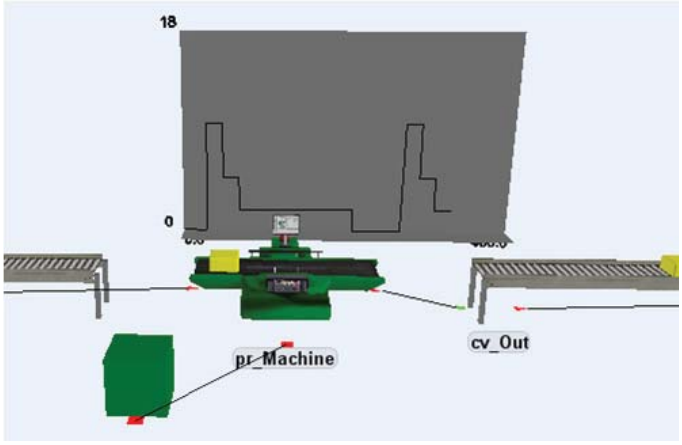

Figure 13.2: Machine Startup Control.

phase. In this case, messages are sent within the same object (the machine) and between objects (a controller and the machine).

The controller, represented in Figure 13.2 as a cube (created using a visual tool), uses its triggers to send start and stop messages to the machine. Whenever the machine receives a start-up message, it begins ramping up its speed as seen in the line graph in the background of the figure which shows the machine's speed over time.

In this example, as shown in the timeline in Figure 13.3, the controller is responsible for turning the machine on and off and, after the stop time, starts the machine using a startup ladder that increases speed from 10 seconds per cycle to 2 seconds per cycle. The machine is scheduled to start 30 minutes after the simulation starts. In order to accomplish this, the controller sends a delayed message to itself through its OnReset trigger. After 30 minutes of simulation, the controller sends a message to the machine to start the startup ramp and a delayed message to itself to stop the machine after 180 minutes.

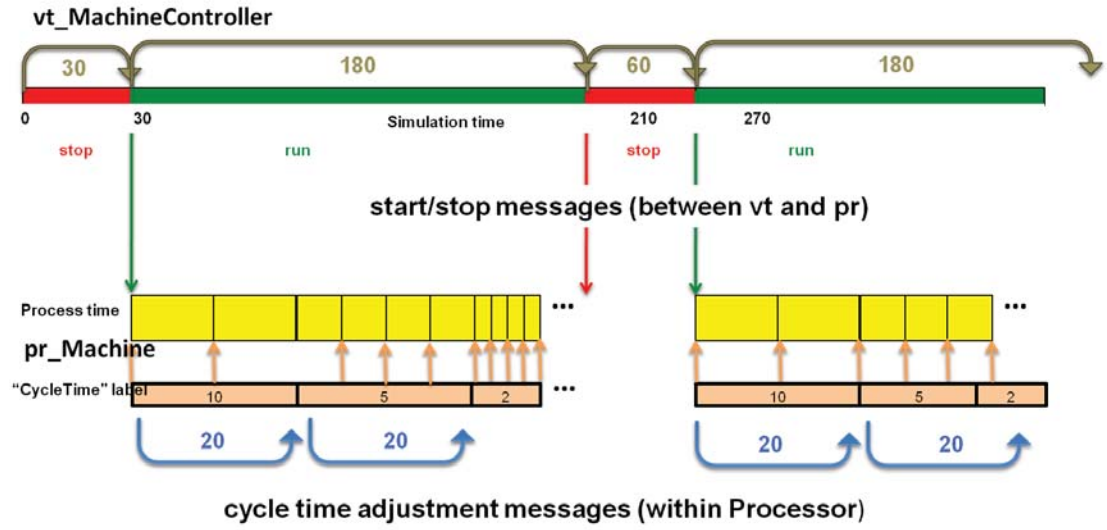

Figure 13.3: Control Timeline.

After three hours of simulated time elapses, the controller receives its message and sends a message to the machine to stop; the controller also sends a message to itself to restart the machine in one hour.

Chapter 17 Managing a Simulation Project

Objective

To describe a set of tools that support the modeling and analysis process, including a simulation project template, a diagram for conceptual model design, and a means to initially size resource requirements.

The previous chapter provided a general outline of how the Simulation Modeling and Analysis (SMA) process is used for analyzing and solving problems with simulation. It highlighted the roles/people and organizational issues involved with the successful use of simulation. This chapter focuses on the technical details of carrying out the SMA process and provides tools that can be used in the process. Those tools include the Object Flow Diagram for conceptual model design, initial resource sizing, and a structure for documenting and managing a simulation project.

Section 17-1 The starting point

An organization that needs to use simulation for analysis must follow a series of steps in order for the simulation to be successful. Building a simulation involves translating reality into a time-based model. As illustrated in Figure 17.1, when individuals look at the actual system, they develop their own conceptual model of how the system operates. Each person may see the system slightly differently or notice different characteristics—all of which may be important. Consequently, a methodology is needed to aide in communications and general understanding.

The Occasional User may not be directly involved with the actual building of the simulation but will be in a position to define what the simulation should accomplish and then possibly carry out simulation runs for analysis. Knowledge of all the steps involved with a simulation project is critical for assuring the project and results are valid.

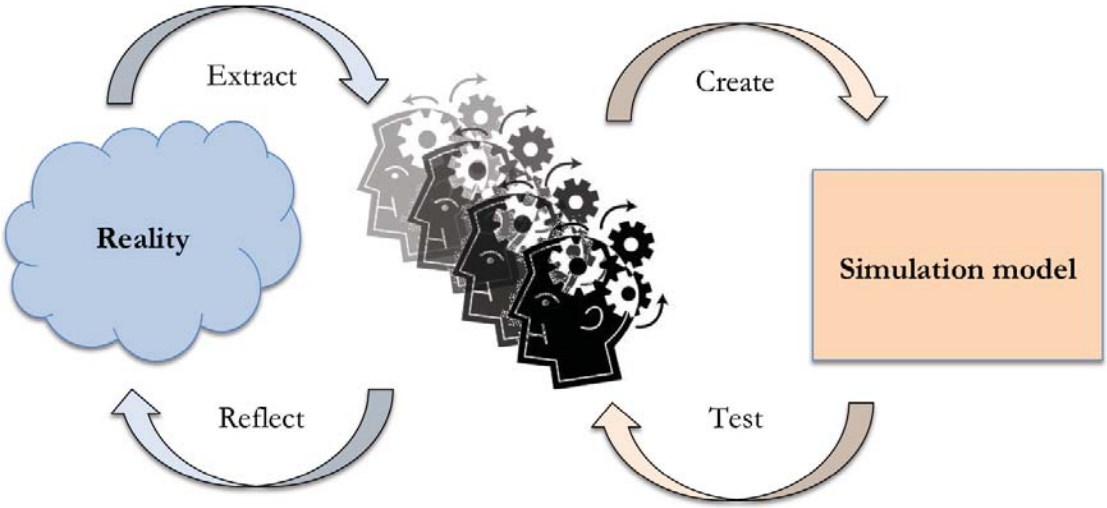

Figure 17.1: A simulation is a representation of stakeholders' reality (Adapted from [4], p. 18).

Simulation project steps

As introduced in Chapter 16, a simulation project is a joint effort between those wanting the simulation and those developing it. The result will only have real value to the people who are involved with the work.

There is a straightforward methodology for defining and carrying out a simulation; this methodology will assure that the resulting model will be of value. It requires working through the steps shown in Figure 17.2; the level of detail for each step will depend on the complexity of the system.

As was also discussed in Chapter 16, the general structure of this process follows Deming's Plan–Do–Study–Act cycle. The first phase is all about planning and understanding. The second phase involves abstracting the system's behavior into model form. The third phase concerns using the simulation model for analysis or study. The final phase is action oriented; that is, it involves gaining insight and information and deciding how to improve system performance based on various experiments conducted with the model.

Activities performed during these steps should be documented using the Simulation Project Template found at the end of this chapter.

Documentation is essential for helping all the individuals working on the simulation to have the same view and understanding of the project. The project template was specifically developed for defining and managing actual simulation projects. The first item in the project template is the project name. The simulation should be given a name that has meaning to the team members and others. The simulation models should contain the name followed by the current version number (e.g., BankSim1).

Since the simulation project is normally approved and carried out for a particular purpose, a summary of the results is usually required. This executive summary is normally limited

1. Define the system and the problem
   - Review existing and proposed facilities and processes
   - Establish goals and objectives
   - Define system and model assumptions
   - Collect data
   - Prepare a simple diagram of system elements

2. Build the simulation model
   - Establish basic parameters for the simulation (e.g., time units)
   - Define and develop user interfaces
   - Layout objects on the screen (simulation surface)
   - Define where and how data will be used
   - Define and implement logic

3. Use the simulation model for analysis
   - Validate and verify the simulation model.
   - Design and run experiments
   - Analyze simulation outputs

4. Use the results to develop an action plan
   - Draw conclusions, present results and recommendations
   - Gain consensus for action plan and implement it.

Figure 17.2: Simulation project steps.

to one page or one slide and focuses on the reason for the simulation and the result of the analysis. The three main parts of the summary are as follows:

- Background (reason for the simulation and objectives)
- Results
- Conclusions and recommendations

Section 17-2 Define the system and the reason to simulate

The first phase of a simulation project involves the critical task of actually defining the project. This definition and the associated material are often referred to as the functional