SIMULATION MODELING AND ANALYSIS

5e

Averill M. Law

Simulation Modeling and Analysis

FIFTH EDITION

Averill M. Law

President
Averill M. Law & Associates, Inc.
Tucson, Arizona, USA
www.averill-law.com

Mc Graw Hill Education

SIMULATION MODELING AND ANALYSIS, FIFTH EDITION

Published by McGraw-Hill Education, 2 Penn Plaza, New York, NY 10121.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 1 0 9 8 7 6 5 4

ISBN 978-0-07-340132-4
MHID 0-07-340132-3

Senior Vice President, Products & Markets: *Kurt L. Strand*
Vice President, General Manager, Products & Markets: *Marty Lange*
Vice President, Content Production & Technology Services: *Kimberly Meriwether David*
Global Publisher: *Raghu Srinivasan*
Development Editor: *Melinda Bilecki*
Marketing Manager: *Heather Wagner*
Director, Content Production: *Terri Schiesl*
Content Project Manager: *Melissa Leick*
Buyer: *Susan K. Culbertson*
Cover Designer: *Studio Montage, St. Louis MO*
Media Project Manager: *Sandy Schnee*
Compositor: *Aptara®, Inc.*
Typeface: *10.5/12 Times*
Printer: *R. R. Donnelley*

Library of Congress Cataloging-in-Publication Data

Law, Averill M.
Simulation modeling and analysis / Averill M. Law, President Averill M. Law & Associates, Inc. Tucson, Arizona, USA, www.averill-law.com. — Fifth edition.
pages cm. — (McGraw-Hill series in industrial engineering and management science)
ISBN 978-0-07-340132-4 (alk. paper)
1. Digital computer simulation. I. Title.
QA76.9.C65L38 2013
003'.3—dc23

2013040962

www.mhhe.com

ABOUT THE AUTHOR

Averill M. Law is President of Averill M. Law & Associates, Inc. (Tucson, Arizona), a company specializing in simulation training, consulting, and software. He was previously Professor of Decision Sciences at the University of Arizona and Associate Professor of Industrial Engineering at the University of Wisconsin–Madison. He has a Ph.D. and an M.S. in industrial engineering and operations research from the University of California at Berkeley, an M.A. in mathematics from California State University at Long Beach, and a B.S. in mathematics from Pennsylvania State University.

Dr. Law has presented more than 525 simulation and statistics short courses in 19 countries, including onsite seminars for ALCOA, AT&T, Boeing, Caterpillar, Coca-Cola, CSX, Defence Research and Development Canada, GE, GM, IBM, Intel, Lockheed Martin, Los Alamos National Lab, Missile Defense Agency, Motorola, NASA, National Security Agency, NATO (Netherlands), Northrop Grumman, Norwegian Defence Research Establishment, Sasol Technology (South Africa), 3M, Time Warner, UPS, U.S. Air Force, U.S. Army, U.S. Forces Korea, U.S. Navy, Verizon, Whirlpool, and Xerox. He has been a simulation consultant to organizations such as Accenture, Boeing, Booz Allen & Hamilton, ConocoPhillips, Defense Modeling and Simulation Office, Hewlett-Packard, Kaiser Aluminum, Kimberly-Clark, M&M/Mars, SAIC, Sandia National Labs, Swedish Defence Materiel Administration, 3M, Tropicana, U.S. Air Force, U.S. Army, U.S. Marine Corps, U.S. Navy, Veteran's Administration, and Xerox.

He is the developer of the ExpertFit distribution-fitting software, which automates the selection of simulation input probability distributions. ExpertFit is used by more than 2000 organizations worldwide. He also developed the videotapes *Simulation of Manufacturing Systems* and *How to Conduct a Successful Simulation Study*.

Dr. Law was awarded the INFORMS Simulation Society Lifetime Professional Achievement Award in 2009. He is the author (or coauthor) of three books and numerous papers on simulation, operations research, statistics, manufacturing, and communications networks. His article "Statistical Analysis of Simulation Output Data" was the first invited feature paper on simulation to appear in a major research journal, namely, *Operations Research*. His series of papers on the simulation of manufacturing systems won the 1988 Institute of Industrial Engineers' best publication award. During his academic career, the Office of Naval Research supported his simulation research for eight consecutive years. He was President of the INFORMS College on Simulation. He wrote a regular column on simulation for *Industrial Engineering* during 1990 and 1991. He has been the keynote speaker at simulation conferences worldwide.

For Steffi, Heather, Adam, and Brian, and in memory of Sallie and David.

CONTENTS

List of Symbols xiii
Preface xvi

Chapter 1 Basic Simulation Modeling 1

1.1 The Nature of Simulation 1
1.2 Systems, Models, and Simulation 3
1.3 Discrete-Event Simulation 6
1.3.1 Time-Advance Mechanisms 7
1.3.2 Components and Organization of a Discrete-Event Simulation Model 9
1.4 Simulation of a Single-Server Queueing System 12
1.4.1 Problem Statement 12
1.4.2 Intuitive Explanation 18
1.4.3 Program Organization and Logic 27
1.4.4 C Program 32
1.4.5 Simulation Output and Discussion 39
1.4.6 Alternative Stopping Rules 41
1.4.7 Determining the Events and Variables 45
1.5 Simulation of an Inventory System 48
1.5.1 Problem Statement 48
1.5.2 Program Organization and Logic 50
1.5.3 C Program 53
1.5.4 Simulation Output and Discussion 60
1.6 Parallel/Distributed Simulation and the High Level Architecture 61
1.6.1 Parallel Simulation 62
1.6.2 Distributed Simulation and the High Level Architecture 64
1.7 Steps in a Sound Simulation Study 66
1.8 Advantages, Disadvantages, and Pitfalls of Simulation 70

Appendix 1A: Fixed-Increment Time Advance 72
Appendix 1B: A Primer on Queueing Systems 73
1B.1 Components of a Queueing System 74
1B.2 Notation for Queueing Systems 74
1B.3 Measures of Performance for Queueing Systems 75

Problems 78

Chapter 2 Modeling Complex Systems 85

2.1 Introduction 85
2.2 List Processing in Simulation 86
2.2.1 Approaches to Storing Lists in a Computer 86
2.2.2 Linked Storage Allocation 87
2.3 A Simple Simulation Language: simlib 93
2.4 Single-Server Queueing Simulation with simlib 102
2.4.1 Problem Statement 102
2.4.2 simlib Program 102
2.4.3 Simulation Output and Discussion 107
2.5 Time-Shared Computer Model 108
2.5.1 Problem Statement 108
2.5.2 simlib Program 109
2.5.3 Simulation Output and Discussion 117
2.6 Multiteller Bank with Jockeying 120
2.6.1 Problem Statement 120
2.6.2 simlib Program 121
2.6.3 Simulation Output and Discussion 131
2.7 Job-Shop Model 134
2.7.1 Problem Statement 134
2.7.2 simlib Program 136
2.7.3 Simulation Output and Discussion 147
2.8 Efficient Event-List Management 149
Appendix 2A: C Code for simlib 150
Problems 163

Chapter 3 Simulation Software 181

3.1 Introduction 181
3.2 Comparison of Simulation Packages with Programming Languages 182
3.3 Classification of Simulation Software 183
3.3.1 General-Purpose vs. Application-Oriented Simulation Packages 183
3.3.2 Modeling Approaches 183
3.3.3 Common Modeling Elements 186
3.4 Desirable Software Features 186
3.4.1 General Capabilities 187
3.4.2 Hardware and Software Requirements 189
3.4.3 Animation and Dynamic Graphics 189
3.4.4 Statistical Capabilities 190
3.4.5 Customer Support and Documentation 192
3.4.6 Output Reports and Graphics 193

3.5 General-Purpose Simulation Packages 193
3.5.1 Arena 193
3.5.2 ExtendSim 198
3.5.3 Simio 206
3.5.4 Other General-Purpose Simulation Packages 212
3.6 Object-Oriented Simulation 212
3.7 Examples of Application-Oriented Simulation Packages 213

Chapter 4 Review of Basic Probability and Statistics 214
4.1 Introduction 214
4.2 Random Variables and Their Properties 214
4.3 Simulation Output Data and Stochastic Processes 226
4.4 Estimation of Means, Variances, and Correlations 229
4.5 Confidence Intervals and Hypothesis Tests for the Mean 233
4.6 The Strong Law of Large Numbers 240
4.7 The Danger of Replacing a Probability Distribution by its Mean 241
Appendix 4A: Comments on Covariance-Stationary Processes 241
Problems 242

Chapter 5 Building Valid, Credible, and Appropriately Detailed Simulation Models 246
5.1 Introduction and Definitions 246
5.2 Guidelines for Determining the Level of Model Detail 249
5.3 Verification of Simulation Computer Programs 251
5.4 Techniques for Increasing Model Validity and Credibility 255
5.4.1 Collect High-Quality Information and Data on the System 256
5.4.2 Interact with the Manager on a Regular Basis 257
5.4.3 Maintain a Written Assumptions Document and Perform a Structured Walk-Through 258
5.4.4 Validate Components of the Model by Using Quantitative Techniques 260
5.4.5 Validate the Output from the Overall Simulation Model 262
5.4.6 Animation 268
5.5 Management's Role in the Simulation Process 269
5.6 Statistical Procedures for Comparing Real-World Observations and Simulation Output Data 269
5.6.1 Inspection Approach 270
5.6.2 Confidence-Interval Approach Based on Independent Data 273
5.6.3 Time-Series Approaches 276
5.6.4 Other Approaches 277
Problems 277

Chapter 6 Selecting Input Probability Distributions 279

6.1 Introduction 279
6.2 Useful Probability Distributions 285
6.2.1 Parameterization of Continuous Distributions 285
6.2.2 Continuous Distributions 286
6.2.3 Discrete Distributions 305
6.2.4 Empirical Distributions 305
6.3 Techniques for Assessing Sample Independence 316
6.4 Activity I: Hypothesizing Families of Distributions 319
6.4.1 Summary Statistics 320
6.4.2 Histograms 322
6.4.3 Quantile Summaries and Box Plots 324
6.5 Activity II: Estimation of Parameters 330
6.6 Activity III: Determining How Representative the Fitted Distributions Are 334
6.6.1 Heuristic Procedures 335
6.6.2 Goodness-of-Fit Tests 344
6.7 The ExpertFit Software and an Extended Example 359
6.8 Shifted and Truncated Distributions 364
6.9 Bézier Distributions 366
6.10 Specifying Multivariate Distributions, Correlations, and Stochastic Processes 367
6.10.1 Specifying Multivariate Distributions 368
6.10.2 Specifying Arbitrary Marginal Distributions and Correlations 372
6.10.3 Specifying Stochastic Processes 373
6.11 Selecting a Distribution in the Absence of Data 375
6.12 Models of Arrival Processes 380
6.12.1 Poisson Processes 380
6.12.2 Nonstationary Poisson Processes 381
6.12.3 Batch Arrivals 384
6.13 Assessing the Homogeneity of Different Data Sets 385

Appendix 6A: Tables of MLEs for the Gamma and Beta Distributions 386

Problems 389

Chapter 7 Random-Number Generators 393

7.1 Introduction 393
7.2 Linear Congruential Generators 397
7.2.1 Mixed Generators 399
7.2.2 Multiplicative Generators 400
7.3 Other Kinds of Generators 402
7.3.1 More General Congruences 402

7.3.2 Composite Generators 403
7.3.3 Feedback Shift Register Generators 405
7.4 Testing Random-Number Generators 409
7.4.1 Empirical Tests 409
7.4.2 Theoretical Tests 414
7.4.3 Some General Observations on Testing 418

Appendix 7A: Portable C Code for a PMMLCG 419
Appendix 7B: Portable C Code for a Combined MRG 421

Problems 423

Chapter 8 Generating Random Variates 426
8.1 Introduction 426
8.2 General Approaches to Generating Random Variates 428
8.2.1 Inverse Transform 428
8.2.2 Composition 437
8.2.3 Convolution 440
8.2.4 Acceptance-Rejection 441
8.2.5 Ratio of Uniforms 448
8.2.6 Special Properties 450
8.3 Generating Continuous Random Variates 451
8.3.1 Uniform 452
8.3.2 Exponential 452
8.3.3 *m*-Erlang 453
8.3.4 Gamma 453
8.3.5 Weibull 456
8.3.6 Normal 457
8.3.7 Lognormal 458
8.3.8 Beta 458
8.3.9 Pearson Type V 459
8.3.10 Pearson Type VI 460
8.3.11 Log-Logistic 460
8.3.12 Johnson Bounded 460
8.3.13 Johnson Unbounded 461
8.3.14 Bézier 461
8.3.15 Triangular 461
8.3.16 Empirical Distributions 462
8.4 Generating Discrete Random Variates 463
8.4.1 Bernoulli 464
8.4.2 Discrete Uniform 464
8.4.3 Arbitrary Discrete Distribution 464
8.4.4 Binomial 469
8.4.5 Geometric 469
8.4.6 Negative Binomial 469
8.4.7 Poisson 470

8.5 Generating Random Vectors, Correlated Random Variates, and Stochastic Processes 470
8.5.1 Using Conditional Distributions 471
8.5.2 Multivariate Normal and Multivariate Lognormal 472
8.5.3 Correlated Gamma Random Variates 473
8.5.4 Generating from Multivariate Families 474
8.5.5 Generating Random Vectors with Arbitrarily Specified Marginal Distributions and Correlations 474
8.5.6 Generating Stochastic Processes 475
8.6 Generating Arrival Processes 476
8.6.1 Poisson Processes 476
8.6.2 Nonstationary Poisson Processes 477
8.6.3 Batch Arrivals 481
Appendix 8A: Validity of the Acceptance-Rejection Method 481
Appendix 8B: Setup for the Alias Method 482
Problems 483

Chapter 9 Output Data Analysis for a Single System 488
9.1 Introduction 488
9.2 Transient and Steady-State Behavior of a Stochastic Process 491
9.3 Types of Simulations with Regard to Output Analysis 493
9.4 Statistical Analysis for Terminating Simulations 497
9.4.1 Estimating Means 498
9.4.2 Estimating Other Measures of Performance 507
9.4.3 Choosing Initial Conditions 510
9.5 Statistical Analysis for Steady-State Parameters 511
9.5.1 The Problem of the Initial Transient 511
9.5.2 Replication/Deletion Approach for Means 523
9.5.3 Other Approaches for Means 526
9.5.4 Estimating Other Measures of Performance 540
9.6 Statistical Analysis for Steady-State Cycle Parameters 542
9.7 Multiple Measures of Performance 545
9.8 Time Plots of Important Variables 548
Appendix 9A: Ratios of Expectations and Jackknife Estimators 550
Problems 551

Chapter 10 Comparing Alternative System Configurations 556
10.1 Introduction 556
10.2 Confidence Intervals for the Difference between the Expected Responses of Two Systems 560
10.2.1 A Paired-*t* Confidence Interval 560

10.2.2 A Modified Two-Sample-t Confidence Interval 562
10.2.3 Contrasting the Two Methods 563
10.2.4 Comparisons Based on Steady-State Measures of Performance 563
10.3 Confidence Intervals for Comparing More than Two Systems 565
10.3.1 Comparisons with a Standard 566
10.3.2 All Pairwise Comparisons 568
10.3.3 Multiple Comparisons with the Best 569
10.4 Ranking and Selection 569
10.4.1 Selecting the Best of k Systems 570
10.4.2 Selecting a Subset of Size m Containing the Best of k Systems 576
10.4.3 Additional Problems and Methods 577
Appendix 10A: Validity of the Selection Procedures 582
Appendix 10B: Constants for the Selection Procedures 583
Problems 584

Chapter 11 Variance-Reduction Techniques 587
11.1 Introduction 587
11.2 Common Random Numbers 588
11.2.1 Rationale 589
11.2.2 Applicability 590
11.2.3 Synchronization 592
11.2.4 Some Examples 596
11.3 Antithetic Variates 604
11.4 Control Variates 610
11.5 Indirect Estimation 617
11.6 Conditioning 619
Problems 623

Chapter 12 Experimental Design and Optimization 629
12.1 Introduction 629
12.2 2^k Factorial Designs 632
12.3 2^{k-p} Fractional Factorial Designs 649
12.4 Response Surfaces and Metamodels 656
12.4.1 Introduction and Analysis of the Inventory Model 657
12.4.2 Analysis of the Predator-Prey Model 668
12.4.3 Space-Filling Designs and Kriging 671
12.5 Simulation-Based Optimization 679
12.5.1 Optimum-Seeking Methods 681
12.5.2 Optimum-Seeking Packages Interfaced with Simulation Software 682
Problems 690

Chapter 13 Agent-Based Simulation and System Dynamics 693
13.1 Introduction 693
13.2 Agent-Based Simulation 694
13.2.1 Detailed Examples 699
13.2.2 Time-Advance Mechanisms for ABS 704
13.2.3 Summary of ABS 707
13.3 Continuous Simulation 707
13.3.1 System Dynamics 708
13.4 Combined Discrete-Continuous Simulation 713
13.5 Monte Carlo Simulation 714
13.6 Spreadsheet Simulation 717
Problems 719

Chapter 14 Simulation of Manufacturing Systems website chapter

Appendix 721
References 725
Index 759

LIST OF SYMBOLS

Notation or abbreviation	Page number of definition
A_i	8
ABS	694
AR, ARMA	373
ARTA	374
ASAP3	538
AV	604
A^T	368, 471
Δb	322
Bernoulli(p)	306
beta(α_1, α_2)	295
bin(t, p)	308
$B(\alpha_1, \alpha_2)$	295
$B(t)$	16
C_{ij}	224
C_j	227
CCD	661
CNI	685
Cor	225
Cov	224
CPU	108
CRN	588
cv	320
CV	610
d	76
$\mathcal{DD}$	573
DES	693
$d(n)$	13
$\hat{d}(n)$	13
df	234
D_i	8
DU(i, j)	307
$E(\ )$	222
EAR	374
Erlang	290
expo(β)	287
FIFO	13
FITA	693
$f(x)$	28, 216
$F(x)$	28
$f(x, y)$	221
F^{-1}	325
gamma(α, β)	288
geom(p)	309
GFSR	407
GI/*G*/*s*	75
GPM	676
$h(x)$	322
H_0	238
H_1	238
H&W	537
HLA	64
IID	12
JSB(α_1, α_2, a, b)	301
JSU(α_1, α_2, γ, β)	303
$\mathcal{KN}$	580
$\mathcal{KN}$++	581
$l(\theta)$	331
L	76
$L(\theta)$	330
L&C	536
LCG	397
LFC	576

Notation or abbreviation	Page number of definition
LFSR	406
LHD	672
LIFO	74
LL(α, β)	299
LN(μ, σ^2)	294
$L(t)$	75
m	222, 489
MC	685
MCB	569
$M/E_2/1$	75
$M/G/1$	75
$M/M/1$	28, 75
$M/M/2$	75
$M/M/s$	75
MLE	330
MRG	402
MRG32k3a	404
MSCO	64
MSE	512
MSER	520
MT19937	408
N(μ, σ^2)	292
N(0, 1)	293
$\mathrm{N}_d(\boldsymbol{\mu}, \Sigma)$	370
NC	685
negbin(s, p)	311
NETA	693
$\mathcal{NM}$	573
NORTA	474
$\mathcal{NSGS}$	576
$\mathcal{OCBA}$	577
PMMLCG	400
$p(x)$	215
$p(x, y)$	220
$P(\ )$	215
Pareto(c, α_2)	389
Poisson(λ)	312
PT5(α, β)	297
PT6(α_1, α_2, β)	298
Q	76
$q(n)$	14
$\hat{q}(n)$	14
$Q(t)$	14
$\mathcal{R}$	573
RTI	64
SBatch	538
SFD	671
Skart	539
(s, S)	48
S_i	8
$S^2(n)$	229
SME	68
t_i	8
$t_{n-1,1-\alpha/2}$	235
$T(n)$	14
TGFSR	408
triang(a, b, m)	304
$u(n)$	16
$\hat{u}(n)$	16
U	28
U(a, b)	286, 714
U(0, 1)	28, 286
Var( )	222
VARTA	374
VIP	578
VRT	587
WASSP	539
Weibull(α, β)	290
WELL	409
w.p.	48
w	76
$w(n)$	40
$\hat{w}(n)$	41
$\tilde{w}(n)$	41
W_i	41
x_q	325
$x_{0.5}$	222
$\mathbf{x}$	369
$\mathbf{X}$	368
$\mathbf{X}_k$	369
$X_{(i)}$	313
$\bar{X}(n)$	229
$\bar{Y}_i(w)$	514
$z_{1-\alpha/2}$	233
α	233, 238, 286
β	238, 285, 503

Notation or abbreviation	Page number of definition
γ	285, 504
$\Gamma(\alpha)$	288
ζ	274, 560
λ	74, 381
$\lambda(t)$	382
$\Lambda(t)$	382
μ	222
$\boldsymbol{\mu}$, $\hat{\boldsymbol{\mu}}$	370
ν	237, 495
ρ	75
ρ_{ij}	225
ρ_j	227
σ	224
σ^2	222
Σ	370
$\hat{\Sigma}$	370
$\Phi(z)$	233
$\chi^2_{k-1,1-\alpha}$	347
$\Psi(\hat{\alpha})$	289
ω	74
$\hat{}$	13
$\approx$	233
$\in$	17
$\sim$	286
$\xrightarrow{\mathcal{D}}$	332
$\binom{t}{x}$	308
$\lfloor x \rfloor$	307
$\lceil x \rceil$	424
$\{\}$	215

PREFACE

The goal of this fifth edition of *Simulation Modeling and Analysis* remains the same as that for the first four editions: to give a comprehensive and state-of-the-art treatment of all the important aspects of a simulation study, including modeling, simulation software, model verification and validation, input modeling, random-number generators, generating random variates and processes, statistical design and analysis of simulation experiments, and to highlight major application areas such as manufacturing. The book strives to motivate intuition about simulation and modeling, as well as to present them in a technically correct yet clear manner. There are many examples and problems throughout, as well as extensive references to the simulation and related literature for further study.

The book can serve as the primary text for a variety of courses, for example

- A first course in simulation at the junior, senior, or beginning-graduate-student level in engineering, manufacturing, business, or computer science (Chaps. 1 through 4 and parts of Chaps. 5 through 9 and 13). At the end of such a course, the student will be prepared to carry out complete and effective simulation studies, and to take advanced simulation courses.
- A second course in simulation for graduate students in any of the above disciplines (most of Chaps. 5 through 12). After completing this course, the student should be familiar with the more advanced methodological issues involved in a simulation study, and should be prepared to understand and conduct simulation research.
- An introduction to simulation as part of a general course in operations research or management science (parts of Chaps. 1, 3, 5, 6, 9, and 13).

For instructors who have adopted the book for use in a course, I have made available for download from the website www.mhhe.com/law a number of teaching support materials. These include a comprehensive set of solutions to the Problems and all the computer code for the simulation models and random-number generators in Chaps. 1, 2, and 7. Adopting instructors should contact their local McGraw-Hill representative for login identification and a password to gain access to the material on this site; local representatives can be identified by calling 1-800-338-3987 or by using the representative locator at www.mhhe.com.

The book can also serve as a definitive reference for simulation practitioners and researchers. To this end I have included a detailed discussion of many practical examples gleaned in part from my own experiences and consulting projects. I have also made major efforts to link subjects to the relevant research literature, both in print and on the web, and to keep this material up to date. Prerequisites for understanding the book are knowledge of basic calculus-based probability and statistics (although I give a review of these topics in Chap. 4) and some experience with computing. For Chaps. 1 and 2 the reader should also be familiar with a general-purpose programming language such as C. Occasionally I will also make use of a small amount of linear algebra or matrix theory. More advanced or technically difficult material is located in starred sections or in appendixes to chapters. At the beginning of each chapter, I suggest sections for a first reading of that chapter.

I have made numerous changes and additions to the fourth edition of the book to arrive at this fifth edition, but the organization has remained mostly the same. I have moved the material on other types of simulation from Chap. 1 to a new Chap. 13, which is discussed below. Chapter 2 on modeling complex systems has been updated to reflect the latest research on efficient event-list management. Chapter 3 has been rewritten and expanded to reflect the current state of the art in simulation software. A common example is now given in three of the leading general-purpose simulation packages. The discussion of confidence intervals and hypothesis tests in Chap. 4 has been greatly enhanced, making the chapter a much more self-contained treatment of the basic probability and statistics needed for the remainder of the book. Chapter 5 makes clearer the distinction between validating and calibrating a model, which is often misunderstood. For Chap. 6 on input modeling, the latest developments in accounting for input-model uncertainty and in modeling arrival processes are discussed. Chapter 7 provides recommendations on the best-available random-number generators. Chapter 8 on generating random variates and processes has only had minor updates. Many of the statistical design-and-analysis methods of Chaps. 9 through 12 have been expanded and updated extensively to reflect current practice and recent research. In particular, Chap. 9 contains a comprehensive discussion of the latest fixed-sample-size and sequential methods for estimating the steady-state mean of a simulated system. The discussion of ranking-and-selection procedures in Chap. 10 has been expanded to include newer and more efficient methods that are not based on the classical indifference-zone approach. Chapter 11 on variance-reduction techniques has only had minor changes. In Chap. 12, I give a much more comprehensive and self-contained discussion of design of experiments and metamodeling, with a particular emphasis on what designs and metamodels to use specifically for simulation modeling. The discussion of simulating manufacturing systems is now in a new Chap. 14, which is available on the book's website www.mhhe.com/law, rather than in the book itself. It has been brought up to date in terms of the latest simulation-software packages and uses of simulation for manufacturing applications. There is a *new* Chap. 13 that discusses agent-based simulation and system dynamics, as well as other types of simulation that were previously discussed in Chap. 1 of the fourth edition. A student version of the ExpertFit distribution-fitting software is now available on the book's website; it can be used to analyze the data sets corresponding to the examples and problems in Chap. 6. The references for all the chapters are collected together at the end of the book, to make this material more compact and convenient to the reader. A large and thorough subject index enhances the book's value as a reference.

This text is available as an eBook at www.CourseSmart.com. At CourseSmart you can take advantage of significant savings off the cost of a print textbook, reduce their impact on the environment, and gain access to powerful web tools for learning. CourseSmart eBooks can be viewed online or downloaded to a computer. The eBooks allow readers to do full text searches, add highlighting and notes, and share notes with others. CourseSmart has the largest selection of eBooks available anywhere. Visit www.CourseSmart.com to learn more and to try a sample chapter.

I would first like to thank my former coauthor David Kelton for his numerous contributions to the first three editions of the book. The formal reviewers for the fifth edition were Christos Alexopoulos (Georgia Institute of Technology), Russell Barton (Pennsylvania State University), Chun-Hung Chen (George Mason University), Shane Henderson (Cornell University), Jack Kleijnen (Tilberg University), Pierre L'Ecuyer (Université de Montréal), Charles Macal (Argonne National Lab), Michael North (Argonne National Lab), and Douglas Samuelson (InfoLogix). They each read one new or significantly changed chapter in great detail and made many valuable suggestions. Knowing that I will certainly inadvertently commit grievous errors of omission, I would nonetheless like to thank the following individuals for their help in various ways: Wayne Adams, Mark Anderson, Sigrun Andradóttir, Jay April, Robert Axtell, Emmett Beeker, Marco Better, Edmund Bitinas, A. J. Bobo, Andrei Borshchev, Nathanael Brown, John Carson, Loren Cobb, Eric Frisco, David Galligan, Nigel Gilbert, Fred Glover, David Goldsman, Daniel Green, Charles Harrell, Thomas Hayson, James Henriksen, Raymond Hill, Kathryn Hoad, Terril Hurst, Andrew Ilachinski, Jeffrey Joines, Harry King, David Krahl, Emily Lada, Michael Lauren, Steffi Law, Thomas Lucas, Gregory McIntosh, Janet McLeavey, Anup Mokashi, Daniel Muller, Rodney Myers, William Nordgren, Ernie Page, Dennis Pegden, David Peterson, Stuart Robinson, Paul Sanchez, Susan Sanchez, Lee Schruben, David Siebert, Jeffrey Smith, David Sturrock, Ali Tafazzoli, Andrew Waller, Hong Wan, Robert Weber, Preston White, and James Wilson.

Averill M. Law
Tucson, AZ

CHAPTER 1

Basic Simulation Modeling

Recommended sections for a first reading: 1.1 through 1.4 (except 1.4.7), 1.7, 1.8

1.1 THE NATURE OF SIMULATION

This is a book about techniques for using computers to imitate, or *simulate*, the operations of various kinds of real-world facilities or processes. The facility or process of interest is usually called a *system*, and in order to study it scientifically we often have to make a set of assumptions about how it works. These assumptions, which usually take the form of mathematical or logical relationships, constitute a *model* that is used to try to gain some understanding of how the corresponding system behaves.

If the relationships that compose the model are simple enough, it may be possible to use mathematical methods (such as algebra, calculus, or probability theory) to obtain *exact* information on questions of interest; this is called an *analytic* solution. However, most real-world systems are too complex to allow realistic models to be evaluated analytically, and these models must be studied by means of simulation. In a *simulation* we use a computer to evaluate a model *numerically*, and data are gathered in order to *estimate* the desired true characteristics of the model.

As an example of the use of simulation, consider a manufacturing company that is contemplating building a large extension on to one of its plants but is not sure if the potential gain in productivity would justify the construction cost. It certainly would not be cost-effective to build the extension and then remove it later if it does not work out. However, a careful simulation study could shed some light on the question by simulating the operation of the plant as it currently exists and as it *would* be *if* the plant were expanded.

Application areas for simulation are numerous and diverse. Below is a list of some particular kinds of problems for which simulation has been found to be a useful and powerful tool:

- Designing and analyzing manufacturing systems
- Evaluating military weapons systems or their logistics requirements
- Determining hardware requirements or protocols for communications networks
- Determining hardware and software requirements for a computer system
- Designing and operating transportation systems such as airports, freeways, ports, and subways
- Evaluating designs for service organizations such as call centers, fast-food restaurants, hospitals, and post offices
- Reengineering of business processes
- Analyzing supply chains
- Determining ordering policies for an inventory system
- Analyzing mining operations

Simulation is one of the most widely used operations-research and management-science techniques, if not *the* most widely used. One indication of this is the Winter Simulation Conference, which attracts 600 to 800 people every year. In addition, there are several other simulation conferences that often have more than 100 participants per year.

There are also several surveys related to the use of operations-research techniques. For example, Lane, Mansour, and Harpell (1993) reported from a longitudinal study, spanning 1973 through 1988, that simulation was consistently ranked as one of the three most important "operations-research techniques." The other two were "math programming" (a catch-all term that includes many individual techniques such as linear programming, nonlinear programming, etc.) and "statistics" (which is not an operations-research technique per se). Gupta (1997) analyzed 1294 papers from the journal *Interfaces* (one of the leading journals dealing with applications of operations research) from 1970 through 1992, and found that simulation was second only to "math programming" among 13 techniques considered.

There have been, however, several impediments to even wider acceptance and usefulness of simulation. First, models used to study large-scale systems tend to be very complex, and writing computer programs to execute them can be an arduous task indeed. This task has been made much easier in recent years by the development of excellent software products that automatically provide many of the features needed to "program" a simulation model. A second problem with simulation of complex systems is that a large amount of computer time is sometimes required. However, this difficulty has become much less severe as computers become faster and cheaper. Finally, there appears to be an unfortunate impression that simulation is just an exercise in computer programming, albeit a complicated one. Consequently, many simulation "studies" have been composed of heuristic model building, programming, and a single run of the program to obtain "the answer." We fear that this attitude, which neglects the important issue of how a properly coded model should be used to make inferences about the system of interest, has doubtless led to erroneous conclusions being drawn from many simulation studies. These questions of simulation *methodology*, which are largely independent of the software and hardware used, form an integral part of the latter chapters of this book.

Perspectives on the historical evolution of simulation modeling may be found in Nance and Sargent (2002).

In the remainder of this chapter (as well as in Chap. 2) we discuss systems and models in considerably greater detail and then show how to write computer programs in a general-purpose language to simulate systems of varying degrees of complexity. All of the computer code shown in this chapter can be downloaded from www.mhhe.com/law.

1.2
SYSTEMS, MODELS, AND SIMULATION

A *system* is defined to be a collection of entities, e.g., people or machines, that act and interact together toward the accomplishment of some logical end. [This definition was proposed by Schmidt and Taylor (1970).] In practice, what is meant by "the system" depends on the objectives of a particular study. The collection of entities that comprise a system for one study might be only a subset of the overall system for another. For example, if one wants to study a bank to determine the number of tellers needed to provide adequate service for customers who want just to cash a check or make a savings deposit, the system can be defined to be that portion of the bank consisting of the tellers and the customers waiting in line or being served. If, on the other hand, the loan officer and the safe-deposit boxes are to be included, the definition of the system must be expanded in an obvious way. [See also Fishman (1978, p. 3).] We define the *state* of a system to be that collection of variables necessary to describe a system at a particular time, relative to the objectives of a study. In a study of a bank, examples of possible state variables are the number of busy tellers, the number of customers in the bank, and the time of arrival of each customer in the bank.

We categorize systems to be of two types, discrete and continuous. A *discrete* system is one for which the state variables change instantaneously at separated points in time. A bank is an example of a discrete system, since state variables—e.g., the number of customers in the bank—change only when a customer arrives or when a customer finishes being served and departs. A *continuous* system is one for which the state variables change continuously with respect to time. An airplane moving through the air is an example of a continuous system, since state variables such as position and velocity can change continuously with respect to time. Few systems in practice are wholly discrete or wholly continuous; but since one type of change predominates for most systems, it will usually be possible to classify a system as being either discrete or continuous.

At some point in the lives of most systems, there is a need to study them to try to gain some insight into the relationships among various components, or to predict performance under some new conditions being considered. Figure 1.1 maps out different ways in which a system might be studied.

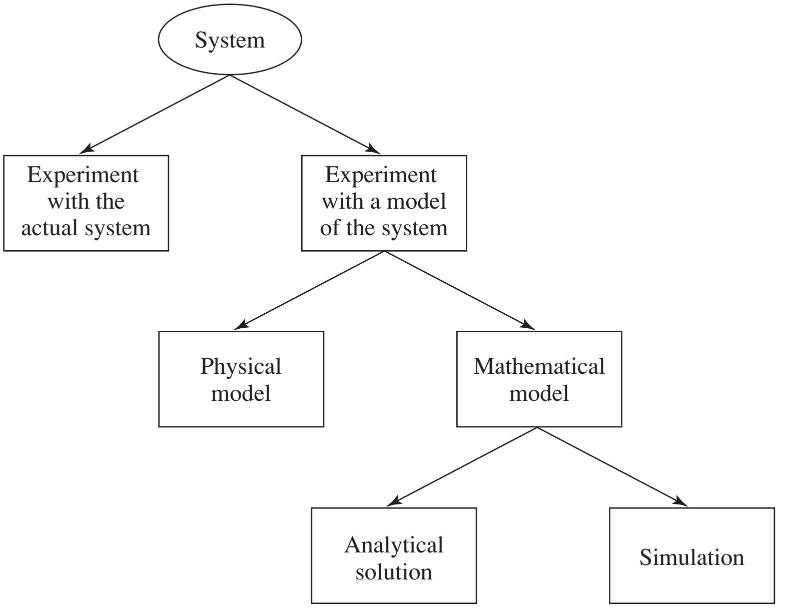

FIGURE 1.1
Ways to study a system.

- *Experiment with the Actual System vs. Experiment with a Model of the System.* If it is possible (and cost-effective) to alter the system physically and then let it operate under the new conditions, it is probably desirable to do so, for in this case there is no question about whether what we study is valid. However, it is rarely feasible to do this, because such an experiment would often be too costly or too disruptive to the system. For example, a bank may be contemplating reducing the number of tellers to decrease costs, but actually trying this could lead to long customer delays and alienation. More graphically, the "system" might not even exist, but we nevertheless want to study it in its various proposed alternative configurations to see how it should be built in the first place; examples of this situation might be a proposed communications network, or a strategic nuclear weapons system. For these reasons, it is usually necessary to build a *model* as a representation of the system and study it as a surrogate for the actual system. When using a model, there is always the question of whether it accurately reflects the system for the purposes of the decisions to be made; this question of model *validity* is taken up in detail in Chap. 5.
- *Physical Model vs. Mathematical Model.* To most people, the word "model" evokes images of clay cars in wind tunnels, cockpits disconnected from their airplanes to be used in pilot training, or miniature supertankers scurrying about in a swimming pool. These are examples of *physical* models (also called *iconic* models), and are not typical of the kinds of models that are usually of interest in operations research and systems analysis. Occasionally, however, it has been found useful to build physical models to study engineering or management

systems; examples include tabletop scale models of material-handling systems, and in at least one case a full-scale physical model of a fast-food restaurant inside a warehouse, complete with full-scale, real (and presumably hungry) humans [see Swart and Donno (1981)]. But the vast majority of models built for such purposes are *mathematical*, representing a system in terms of logical and quantitative relationships that are then manipulated and changed to see how the model reacts, and thus how the system *would* react—*if* the mathematical model is a valid one. Perhaps the simplest example of a mathematical model is the familiar relation $d = rt$, where r is the rate of travel, t is the time spent traveling, and d is the distance traveled. This might provide a valid model in one instance (e.g., a space probe to another planet after it has attained its flight velocity) but a very poor model for other purposes (e.g., rush-hour commuting on congested urban freeways).

- *Analytical Solution vs. Simulation.* Once we have built a mathematical model, it must then be examined to see how it can be used to answer the questions of interest about the system it is supposed to represent. If the model is simple enough, it may be possible to work with its relationships and quantities to get an exact, *analytical* solution. In the $d = rt$ example, if we know the distance to be traveled and the velocity, then we can work with the model to get $t = d/r$ as the time that will be required. This is a very simple, closed-form solution obtainable with just paper and pencil, but some analytical solutions can become extraordinarily complex, requiring vast computing resources; inverting a large nonsparse matrix is a well-known example of a situation in which there is an analytical formula known in principle, but obtaining it numerically in a given instance is far from trivial. If an analytical solution to a mathematical model is available and is computationally efficient, it is usually desirable to study the model in this way rather than via a simulation. However, many systems are highly complex, so that valid mathematical models of them are themselves complex, precluding any possibility of an analytical solution. In this case, the model must be studied by means of *simulation*, i.e., numerically exercising the model for the inputs in question to see how they affect the output measures of performance.

While there may be a small element of truth to pejorative old saws such as "method of last resort" sometimes used to describe simulation, the fact is that we are very quickly led to simulation in most situations, due to the sheer complexity of the systems of interest and of the models necessary to represent them in a valid way.

Given, then, that we have a mathematical model to be studied by means of simulation (henceforth referred to as a *simulation model*), we must then look for particular tools to do this. It is useful for this purpose to classify simulation models along three different dimensions:

- *Static vs. Dynamic Simulation Models.* A *static* simulation model is a representation of a system at a particular time, or one that may be used to represent a system in which time simply plays no role; examples of static simulations are certain Monte Carlo models, discussed in Sec. 13.5. On the other hand, a *dynamic* simulation model represents a system as it evolves over time, such as a conveyor system in a factory.

- *Deterministic vs. Stochastic Simulation Models*. If a simulation model does not contain any probabilistic (i.e., random) components, it is called *deterministic*; a complicated (and analytically intractable) system of differential equations describing a chemical reaction might be such a model. In deterministic models, the output is "determined" once the set of input quantities and relationships in the model have been specified, even though it might take a lot of computer time to evaluate what it is. Many systems, however, must be modeled as having at least some random input components, and these give rise to *stochastic* simulation models. (For an example of the danger of ignoring randomness in modeling a system, see Sec. 4.7.) Most queueing and inventory systems are modeled stochastically. Stochastic simulation models produce output that is itself random, and must therefore be treated as only an estimate of the true characteristics of the model; this is one of the main disadvantages of simulation (see Sec. 1.8) and is dealt with in Chaps. 9 through 12 of this book.
- *Continuous vs. Discrete Simulation Models.* Loosely speaking, we define *discrete* and *continuous* simulation models analogously to the way discrete and continuous systems were defined above. More precise definitions of discrete (event) simulation and continuous simulation are given in Secs. 1.3 and 13.3, respectively. It should be mentioned that a discrete model is not always used to model a discrete system, and vice versa. The decision whether to use a discrete or a continuous model for a particular system depends on the specific objectives of the study. For example, a model of traffic flow on a freeway would be discrete if the characteristics and movement of individual cars are important. Alternatively, if the cars can be treated "in the aggregate," the flow of traffic can be described by differential equations in a continuous model. More discussion on this issue can be found in Sec. 5.2, and in particular in Example 5.2.

The simulation models we consider in the remainder of this book, except for those in Chap. 13, will be discrete, dynamic, and stochastic and will henceforth be called *discrete-event simulation models*. (Since deterministic models are a special case of stochastic models, the restriction to stochastic models involves no loss of generality.)

1.3
DISCRETE-EVENT SIMULATION

Discrete-event simulation concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. (In more mathematical terms, we might say that the system can change at only a *countable* number of points in time.) These points in time are the ones at which an event occurs, where an *event* is defined as an instantaneous occurrence that may change the state of the system. Although discrete-event simulation could conceptually be done by hand calculations, the amount of data that must be stored and manipulated for most real-world systems dictates that discrete-event simulations be done on a digital computer. (In Sec. 1.4.2 we carry out a small hand simulation, merely to illustrate the logic involved.)

EXAMPLE 1.1. Consider a service facility with a single server—e.g., a one-operator barbershop or an information desk at an airport—for which we would like to estimate the (expected) average delay in queue (line) of arriving customers, where the delay in queue of a customer is the length of the time interval from the instant of his arrival at the facility to the instant he begins being served. For the objective of estimating the average delay of a customer, the state variables for a discrete-event simulation model of the facility would be the status of the server, i.e., either idle or busy, the number of customers waiting in queue to be served (if any), and the time of arrival of each person waiting in queue. The status of the server is needed to determine, upon a customer's arrival, whether the customer can be served immediately or must join the end of the queue. When the server completes serving a customer, the number of customers in the queue is used to determine whether the server will become idle or begin serving the first customer in the queue. The time of arrival of a customer is needed to compute his delay in queue, which is the time he begins being served (which will be known) minus his time of arrival. There are two types of events for this system: the arrival of a customer and the completion of service for a customer, which results in the customer's departure. An arrival is an event since it causes the (state variable) server status to change from idle to busy or the (state variable) number of customers in the queue to increase by 1. Correspondingly, a departure is an event because it causes the server status to change from busy to idle or the number of customers in the queue to decrease by 1. We show in detail how to build a discrete-event simulation model of this single-server queueing system in Sec. 1.4.

In the above example both types of events actually changed the state of the system, but in some discrete-event simulation models events are used for purposes that do not actually effect such a change. For example, an event might be used to schedule the end of a simulation run at a particular time (see Sec. 1.4.6) or to schedule a decision about a system's operation at a particular time (see Sec. 1.5) and might not actually result in a change in the state of the system. This is why we originally said that an event *may* change the state of a system.

1.3.1 Time-Advance Mechanisms

Because of the dynamic nature of discrete-event simulation models, we must keep track of the current value of simulated time as the simulation proceeds, and we also need a mechanism to advance simulated time from one value to another. We call the variable in a simulation model that gives the current value of simulated time the *simulation clock*. The unit of time for the simulation clock is never stated explicitly when a model is written in a general-purpose language such as C, and it is assumed to be in the same units as the input parameters. Also, there is generally no relationship between simulated time and the time needed to run a simulation on the computer.

Historically, two principal approaches have been suggested for advancing the simulation clock: *next-event time advance* and *fixed-increment time advance.* Since the first approach is used by all major simulation software and by most people programming their model in a general-purpose language, and since the second is a special case of the first, we shall use the next-event time-advance approach for all discrete-event simulation models discussed in this book. A brief discussion of fixed-increment time advance is given in App. 1A (at the end of this chapter).

With the next-event time-advance approach, the simulation clock is initialized to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the time of occurrence of the *most imminent* (first) of these future events, at which point the state of the system is updated to account for the fact that an event has occurred, and our knowledge of the times of occurrence of future events is also updated. Then the simulation clock is advanced to the time of the (new) most imminent event, the state of the system is updated, and future event times are determined, etc. This process of advancing the simulation clock from one event time to another is continued until eventually some prespecified stopping condition is satisfied. Since all state changes occur only at event times for a discrete-event simulation model, periods of inactivity are skipped over by jumping the clock from event time to event time. (Fixed-increment time advance does not skip over these inactive periods, which can eat up a lot of computer time; see App. 1A.) It should be noted that the successive jumps of the simulation clock are generally variable (or unequal) in size.

EXAMPLE 1.2. We now illustrate in detail the next-event time-advance approach for the single-server queueing system of Example 1.1. We need the following notation:

t_i = time of arrival of the ith customer ($t_0 = 0$)
$A_i = t_i - t_{i-1}$ = interarrival time between $(i - 1)$st and ith arrivals of customers
S_i = time that server actually spends serving ith customer (exclusive of customer's delay in queue)
D_i = delay in queue of ith customer
$c_i = t_i + D_i + S_i$ = time that ith customer completes service and departs
e_i = time of occurrence of ith event of any type (ith value the simulation clock takes on, excluding the value $e_0 = 0$)

Each of these defined quantities will generally be a random variable. Assume that the probability distributions of the interarrival times $A_1, A_2, \ldots$ and the service times $S_1, S_2, \ldots$ are known and have cumulative distribution functions (see Sec. 4.2) denoted by F_A and F_S, respectively. (In general, F_A and F_S would be determined by collecting data from the system of interest and then specifying distributions consistent with these data using the techniques of Chap. 6.) At time $e_0 = 0$ the status of the server is idle, and the time t_1 of the first arrival is determined by generating A_1 from F_A (techniques for generating random observations from a specified distribution are discussed in Chap. 8) and adding it to 0. The simulation clock is then advanced from e_0 to the time of the next (first) event, $e_1 = t_1$. (See Fig. 1.2, where the curved arrows represent advancing the simulation clock.) Since the customer arriving at time t_1 finds the server idle, she immediately enters service and has a delay in queue of $D_1 = 0$ and the status of the server is changed from idle to busy. The time, c_1, when the arriving customer will complete service is computed by generating S_1 from F_S and adding it to t_1. Finally, the time of the second arrival, t_2, is computed as $t_2 = t_1 + A_2$, where A_2 is generated from F_A. If $t_2 < c_1$, as depicted in Fig. 1.2, the simulation clock is advanced from e_1 to the time of the next event, $e_2 = t_2$. (If c_1 were less than t_2, the clock would be advanced from e_1 to c_1.) Since the customer arriving at time t_2 finds the server already busy, the number of customers in the queue is increased from 0 to 1 and the time of arrival of this customer is recorded; however, his service time S_2 is not generated at this time. Also, the time of the third arrival, t_3, is computed as $t_3 = t_2 + A_3$. If $c_1 < t_3$, as depicted in the figure, the simulation clock is advanced from e_2 to the time of the next event, $e_3 = c_1$, where the customer

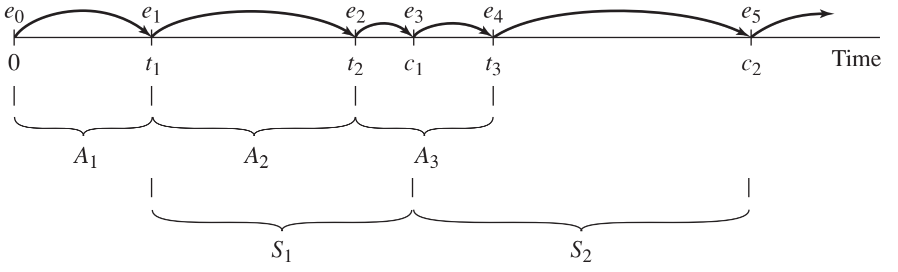

FIGURE 1.2
The next-event time-advance approach illustrated for the single-server queueing system.

completing service departs, the customer in the queue (i.e., the one who arrived at time t_2) begins service and his delay in queue and service-completion time are computed as $D_2 = c_1 - t_2$ and $c_2 = c_1 + S_2$ (S_2 is now generated from F_S), and the number of customers in the queue is decreased from 1 to 0. If $t_3 < c_2$, the simulation clock is advanced from e_3 to the time of the next event, $e_4 = t_3$, etc. The simulation might eventually be terminated when, say, the number of customers whose delays have been observed reaches some specified value.

1.3.2 Components and Organization of a Discrete-Event Simulation Model

Although simulation has been applied to a great diversity of real-world systems, discrete-event simulation models all share a number of common components and there is a logical organization for these components that promotes the programming, debugging, and future changing of a simulation model's computer program. In particular, the following components will be found in most discrete-event simulation models using the next-event time-advance approach programmed in a general-purpose language:

System state: The collection of state variables necessary to describe the system at a particular time
Simulation clock: A variable giving the current value of simulated time
Event list: A list containing the next time when each type of event will occur
Statistical counters: Variables used for storing statistical information about system performance
Initialization routine: A subprogram to initialize the simulation model at time 0
Timing routine: A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur
Event routine: A subprogram that updates the system state when a particular type of event occurs (there is one event routine for each event type)
Library routines: A set of subprograms used to generate random observations from probability distributions that were determined as part of the simulation model