Handbook of Software Reliability Engineering
To my wife C. Felicia Lyu, for her love, understanding, and support throughout this project.
Contents

Contributors  xviii Foreword by Alfred V. Aho  xix Foreword by Richard A. DeMillo  xxi Preface xxiii

Part 1 Technical Foundations

Chapter 1. Introduction  3

1.1 The Need for Reliable Software  3
1.2 Software Reliability Engineering Concepts  5
1.3 Book Overview  8
1.4 Basic Definitions  12
1.5 Technical Areas Related to the Book  19
   1.5.1 Fault Prevention  19
   1.5.2 Fault Removal  20
   1.5.3 Fault Tolerance  20
   1.5.4 Fault/Failure Forecasting  21
   1.5.5 Scope of This Handbook  21
1.6 Summary  22
Problems  22

Chapter 2. Software Reliability and System Reliability 27

2.1 Introduction  27
2.2 The Dependability Concept  28
   2.2.1 Basic Definitions  28
   2.2.2 On the Impairments to Dependability  28
   2.2.3 On the Attributes of Dependability  32
   2.2.4 On the Means for Dependability  33
2.3 Failure Behavior of an X-ware System  35
   2.3.1 Atomic Systems  35
   2.3.2 Systems Made Up of Components  41
2.4 Failure Behavior of an X-ware System with Service Restoration 49
   2.4.1 Characterization of System Behavior  50
   2.4.2 Maintenance Policies  51
Chapter 3. Software Reliability Modeling Survey

3.1 Introduction
3.2 Historical Perspective and Implementation
   3.2.1 Historical Background
   3.2.2 Model Classification Scheme
   3.2.3 Model Limitations and Implementation Issues
3.3 Exponential Failure Time Class of Models
   3.3.1 Jelinski-Moranda De-eutrophication Model
   3.3.2 Nonhomogeneous Poisson Process (NHPP) Model
   3.3.3 Schneidewind's Model
   3.3.4 Musa's Basic Execution Time Model
   3.3.5 Hyperexponential Model
   3.3.6 Others
3.4 Weibull and Gamma Failure Time Class of Models
   3.4.1 Weibull Model
   3.4.2 S-Shaped Reliability Growth Model
3.5 Infinite Failure Category Models
   3.5.1 Duane's Model
   3.5.2 Geometric Model
   3.5.3 Musa-Okumoto Logarithmic Poisson
3.6 Bayesian Models
   3.6.1 Littlewood-Verrall Reliability Growth Model
   3.6.2 Other Bayesian Models
3.7 Model Relationships
   3.7.1 Generalized Exponential Model Class
   3.7.2 Exponential Order Statistic Model Class
3.8 Software Reliability Prediction in Early Phases of the Life Cycle
   3.8.1 Phase-Based Model
   3.8.2 Predicting Software Defects from Ada Designs
   3.8.3 Rome Laboratory Work
3.9 Summary Problems

Chapter 4. Techniques for Prediction Analysis and Recalibration

4.1 Introduction
4.2 Examples of Model Disagreement and Inaccuracy
   4.2.1 Simple Short-Term Predictions
   4.2.2 Longer-Term Predictions
   4.2.3 Model Accuracy Varies from Data Source to Data Source
   4.2.4 Why We Cannot Select the Best Model a Priori
   4.2.5 Discussion: A Possible Way Forward
4.3 Methods of Analyzing Predictive Accuracy
   4.3.1 Basic Ideas: Recursive Comparison of Predictions with Eventual Outcomes
Chapter 5. The Operational Profile

5.1 Introduction 167
5.2 Concepts 168
5.3 Development Procedure 170
  5.3.1 Customer Type List 173
  5.3.2 User Type List 173
  5.3.3 System Mode List 174
  5.3.4 Functional Profile 176
  5.3.5 Operational Profile 183
5.4 Test Selection 194
  5.4.1 Selecting Operations 195
  5.4.2 Regression Test 196
5.5 Special Issues 197
  5.5.1 Indirect Input Variables 197
  5.5.2 Updating the Operational Profile 197
  5.5.3 Distributed Systems 198
5.6 Other Uses 199
5.7 Application to DEFINTY® 200
  5.7.1 Project Description 200
  5.7.2 Development Process Description 200
  5.7.3 Describing Operational Profiles 201
  5.7.4 Implementing Operational Profiles 203
  5.7.5 Conclusion 204
5.8 Application to FASTAR® (FAST Automated Restoration) 204
  5.8.1 System Description 204
  5.8.2 FASTAR: SRE Implementation 206
  5.8.3 FASTAR: SRE Benefits 210
5.9 Application to the Power Quality Resource System (PQRS) 210
  5.9.1 Project Description 210
  5.9.2 Developing the Operational Profile 211
  5.9.3 Testing 213
  5.9.4 Conclusion 214
  5.10 Summary 215
Problems 215
Part 2 Practices and Experiences

Chapter 6. Best Current Practice of SRE 219

6.1 Introduction 219
6.2 Benefits and Approaches to SRE 220
   6.2.1 Importance and Benefits 221
   6.2.2 An SRE Success Story 221
   6.2.3 SRE Costs 222
6.2.4 SRE Activities 223
6.2.5 Implementing SRE Incrementally 223
   6.2.6 Implementing SRE on Existing Projects 224
   6.2.7 Implementing SRE on Short-Cycle Projects 226
6.3 SRE During the Feasibility and Requirements Phase 226
   6.3.1 Feasibility Stage 226
   6.3.2 Requirements Stage 228
6.4 SRE During Design and Implementation Phase 232
   6.4.1 Design Stage 232
   6.4.2 Implementation Stage 233
6.5 SRE During the System Test and Field Trial Phase 235
   6.5.1 Determine Operational Profile 236
   6.5.2 System Test Stage 237
   6.5.3 Field Trial Stage 241
6.6 SRE During the Postdelivery and Maintenance Phase 242
   6.6.1 Project Postrelease Staff Needs 242
   6.6.2 Monitor Field Reliability versus Objectives 243
   6.6.3 Track Customer Satisfaction 245
   6.6.4 Time New Feature Introduction by Monitoring Reliability 245
   6.6.5 Guide Produce and Process Improvement with Reliability Measures 246
6.7 Getting Started with SRE 246
   6.7.1 Prepare Your Organization for SRE 247
   6.7.2 Find More Information or Support 250
   6.7.3 Do an SRE Self-Assessment 250
6.8 Summary 252

Problems 253

Chapter 7. Software Reliability Measurement Experience 255

7.1 Introduction 255
7.2 Measurement Framework 256
   7.2.1 Establishing Software Reliability Requirements 259
   7.2.2 Setting Up a Data Collection Process 266
   7.2.3 Defining Data to Be Collected 267
   7.2.4 Choosing a Preliminary Set of Software Reliability Models 272
   7.2.5 Choosing Reliability Modeling Tools 273
   7.2.6 Model Application and Application Issues 273
   7.2.7 Dealing with Evolving Software 276
   7.2.8 Practical Limits in Modeling Ultrareliability 277
7.3 Project Investigation at JPL 278
   7.3.1 Project Selection and Characterization 278
   7.3.2 Characterization of Available Data 280
   7.3.3 Experimental Results 280
7.4 Investigation at Bellcore 281
7.4.1 Project Characteristics 281
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4.2 Data Collection</td>
<td>284</td>
</tr>
<tr>
<td>7.4.3 Application Results</td>
<td>285</td>
</tr>
<tr>
<td>7.5 Linear Combination of Model Results</td>
<td>289</td>
</tr>
<tr>
<td>7.5.1 Statically Weighted Linear Combinations</td>
<td>290</td>
</tr>
<tr>
<td>7.5.2 Weight Determination Based on Ranking Model Results</td>
<td>290</td>
</tr>
<tr>
<td>7.5.3 Weight Determination Based on Changes in Prequential Likelihood</td>
<td>291</td>
</tr>
<tr>
<td>7.5.4 Modeling Results</td>
<td>292</td>
</tr>
<tr>
<td>7.5.5 Overall Project Results</td>
<td>295</td>
</tr>
<tr>
<td>7.5.6 Extensions and Alternatives</td>
<td>295</td>
</tr>
<tr>
<td>7.5.7 Long-Term Prediction Capability</td>
<td>298</td>
</tr>
<tr>
<td>7.6 Summary</td>
<td>299</td>
</tr>
<tr>
<td>Problems</td>
<td>300</td>
</tr>
</tbody>
</table>

## Chapter 8. Measurement-Based Analysis of Software Reliability 303

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Introduction</td>
<td>303</td>
</tr>
<tr>
<td>8.2 Framework</td>
<td>304</td>
</tr>
<tr>
<td>8.2.1 Overview</td>
<td>304</td>
</tr>
<tr>
<td>8.2.2 Operational versus Development Phase Evaluation</td>
<td>306</td>
</tr>
<tr>
<td>8.2.3 Past Work</td>
<td>306</td>
</tr>
<tr>
<td>8.3 Measurement Techniques</td>
<td>307</td>
</tr>
<tr>
<td>8.3.1 On-Line Machine Logging</td>
<td>308</td>
</tr>
<tr>
<td>8.3.2 Manual Reporting</td>
<td>310</td>
</tr>
<tr>
<td>8.4 Preliminary Analysis of Data</td>
<td>312</td>
</tr>
<tr>
<td>8.4.1 Data Processing</td>
<td>312</td>
</tr>
<tr>
<td>8.4.2 Fault and Error Classification</td>
<td>314</td>
</tr>
<tr>
<td>8.4.3 Error Propagation</td>
<td>317</td>
</tr>
<tr>
<td>8.4.4 Error and Recovery Distributions</td>
<td>320</td>
</tr>
<tr>
<td>8.5 Detailed Analysis of Data</td>
<td>323</td>
</tr>
<tr>
<td>8.5.1 Dependency Analysis</td>
<td>324</td>
</tr>
<tr>
<td>8.5.2 Hardware-Related Software Errors</td>
<td>327</td>
</tr>
<tr>
<td>8.5.3 Evaluation of Software Fault Tolerance</td>
<td>328</td>
</tr>
<tr>
<td>8.5.4 Recurrences</td>
<td>329</td>
</tr>
<tr>
<td>8.6 Model Identification and Analysis of Models</td>
<td>333</td>
</tr>
<tr>
<td>8.6.1 Impact of Failures on Performance</td>
<td>333</td>
</tr>
<tr>
<td>8.6.2 Reliability Modeling in the Operational Phase</td>
<td>335</td>
</tr>
<tr>
<td>8.6.3 Error/Failure/Recovery Model</td>
<td>339</td>
</tr>
<tr>
<td>8.6.4 Multiple-Error Model</td>
<td>344</td>
</tr>
<tr>
<td>8.7 Impact of System Activity</td>
<td>345</td>
</tr>
<tr>
<td>8.7.1 Statistical Models from Measurements</td>
<td>345</td>
</tr>
<tr>
<td>8.7.2 Overall System Behavior Model</td>
<td>348</td>
</tr>
<tr>
<td>8.8 Summary</td>
<td>352</td>
</tr>
<tr>
<td>Problems</td>
<td>353</td>
</tr>
</tbody>
</table>

## Chapter 9. Orthogonal Defect Classification 359

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 Introduction</td>
<td>359</td>
</tr>
<tr>
<td>9.2 Measurement and Software</td>
<td>360</td>
</tr>
<tr>
<td>9.2.1 Software Defects</td>
<td>361</td>
</tr>
<tr>
<td>9.2.2 The Spectrum of Defect Analysis</td>
<td>364</td>
</tr>
<tr>
<td>9.3 Principles of ODC</td>
<td>367</td>
</tr>
<tr>
<td>9.3.1 The Intuition</td>
<td>367</td>
</tr>
<tr>
<td>9.3.2 The Design of Orthogonal Defect Classification</td>
<td>370</td>
</tr>
</tbody>
</table>
Chapter 10. Trend Analysis

10.1 Introduction
10.2 Reliability Growth Characterization
  10.2.1 Definitions of Reliability Growth
  10.2.2 Graphical Interpretation of the Subadditive Property
  10.2.3 Subadditive Property Analysis
  10.2.4 Subadditive Property and Trend Change
  10.2.5 Some Particular Situations
  10.2.6 Summary
10.3 Trend Analysis
  10.3.1 Trend Tests
  10.3.2 Example
  10.3.3 Typical Results That Can Be Drawn from Trend Analyses
  10.3.4 Summary
10.4 Application to Real Systems
  10.4.1 Software of System SS4
  10.4.2 Software of System S27
  10.4.3 Software of System SS1
  10.4.4 Software of System SS2
  10.4.5 SAV
10.5 Extension to Static Analysis
  10.5.1 Static Analysis Conduct
  10.5.2 Application
10.6 Summary Problems

Chapter 11. Field Data Analysis

11.1 Introduction
11.2 Data Collection Principles
  11.2.1 Introduction
  11.2.2 Failures, Faults, and Related Data
  11.2.3 Time
  11.2.4 Usage
  11.2.5 Data Granularity
  11.2.6 Data Maintenance and Validation
11.2.7 Analysis Environment
11.3 Data Analysis Principles
11.3.1 Plots and Graphs
11.3.2 Data Modeling and Diagnostics
11.3.3 Diagnostics for Model Determination
11.3.4 Data Transformations
11.4 Important Topics in Analysis of Field Data
11.4.1 Calendar Time
11.4.2 Usage Time
11.4.3 An Example
11.5 Calendar-Time Reliability Analysis
11.5.1 Case Study (IBM Corporation)
11.5.2 Case Study (Hitachi Software Engineering Company)
11.5.3 Further Examples
11.6 Usage-Based Reliability Analysis
11.6.1 Case Study (Nortel Telecommunication Systems)
11.6.2 Further Examples
11.7 Special Events
11.7.1 Rare Event Models
11.7.2 Case Study (Space Shuttle Flight Software)
11.8 Availability
11.8.1 Introduction
11.8.2 Measuring Availability
11.8.3 Empirical Unavailability
11.8.4 Models
11.9 Summary Problems

Part 3 Emerging Techniques

Chapter 12. Software Metrics for Reliability Assessment

12.1 Introduction
12.2 Static Program Complexity
12.2.1 Software Metrics
12.2.2 A Domain Model of Software Attributes
12.2.3 Principal Components Analysis
12.2.4 The Usage of Metrics
12.2.5 Relative Program Complexity
12.2.6 Software Evolution
12.3 Dynamic Program Complexity
12.3.1 Execution Profile
12.3.2 Functional Complexity
12.3.3 Dynamic Aspects of Functional Complexity
12.3.4 Operational Complexity
12.4 Software Complexity and Software Quality
12.4.1 Overview
12.4.2 The Application and Its Metrics
12.4.3 Multivariate Analysis in Software Quality Control
12.4.4 Fault Prediction Models
12.4.5 Enhancing Predictive Models with Increased Domain Coverage
12.5 Software Reliability Modeling
12.5.1 Reliability Modeling with Software Complexity Metrics
12.5.2 The Incremental Build Problem
Chapter 13. Software Testing and Reliability

13.1 Introduction

13.2 Overview of Software Testing
  13.2.1 Kinds of Software Testing
  13.2.2 Concepts from White-Box and Black-Box Testing

13.3 Operational Profiles
  13.3.1 Difficulties in Estimating the Operational Profile
  13.3.2 Estimating Reliability with Inaccurate Operational Profiles

13.4 Time/Structure-Based Software Reliability Estimation
  13.4.1 Definitions and Terminology
  13.4.2 Basic Assumptions
  13.4.3 Testing Methods and Saturation Effect
  13.4.4 Testing Effort
  13.4.5 Limits of Testing Methods
  13.4.6 Empirical Basis of the Saturation Effect
  13.4.7 Reliability Overestimation due to Saturation
  13.4.8 Incorporating Coverage in Reliability Estimation
  13.4.9 Filtering Failure Data Using Coverage Information
  13.4.10 Selecting the Compression Ratio
  13.4.11 Handling Rare Events

13.5 A Microscopic Model of Software Risk
  13.5.1 A Testing-Based Model of Risk Decay
  13.5.2 Risk Assessment: An Example
  13.5.3 A Simple Risk Computation
  13.5.4 A Risk Browser
  13.5.5 The Risk Model and Software Reliability

13.6 Summary Problems

Chapter 14. Fault-Tolerant Software Reliability Engineering

14.1 Introduction

14.2 Present Status

14.3 Principles and Terminology
  14.3.1 Result Verification
  14.3.2 Redundancy
  14.3.3 Failures and Faults
  14.3.4 Adjudication by Voting
  14.3.5 Tolerance

14.4 Basic Techniques
  14.4.1 Recovery Blocks
  14.4.2 N-Version Programming

14.5 Advanced Techniques
  14.5.1 Consensus Recovery Block
  14.5.2 Acceptance Voting
  14.5.3 N Self-Checking Programming

14.6 Reliability Modeling
  14.6.1 Diversity and Dependence of Failures
  14.6.2 Data-Domain Modeling
  14.6.3 Time-Domain Modeling
Contributors

Sarah Brocklehurst City University, London (CHAP. 4)
Ram Chilarege IBM Watson Research (CHAP. 9)
Mary Donnelly AT&T Bell Laboratories (CHAP. 6)
Joanne Bechta Dugan University of Virginia (CHAP. 15)
Bill Everett AT&T Bell Laboratories (CHAP. 6)
William Farr Naval Surface Warfare Center (CHAP. 3)
Gene Fuoco AT&T Bell Laboratories (CHAP. 5)
Joseph R. Horgan Bellcore (CHAP. 13)
Nancy Irving AT&T Bell Laboratories (CHAP. 5)
Ravi K. Iyer University of Illinois (CHAP. 8)
Wendell Jones BNR Incorporated (CHAP. 11)
Bruce Juhlin US West (CHAP. 5)
Karama Kanoun LAAS-CNRS, Toulouse, France (CHAP. 2, 10)
Nachimuthu Karunanithi Bellcore (CHAP. 17)
Taghi Khoshgoftaar Florida Atlantic University (CHAP. 12)
Diane Kropfl AT&T Bell Laboratories (CHAP. 5)
Jean-Claude Laprie LAAS-CNRS, Toulouse, France (CHAP. 2, 10)
Inhwan Lee Tandem Computers, Inc. (CHAP. 8)
Bev Littlewood City University, London (CHAP. 4)
Michael R. Lyu AT&T Bell Laboratories, Editor (CHAP. 1, 7, 16, APP. B)
Yashwant Malaiya Colorado State University (CHAP. 17)
Aditya P. Mathur Purdue University (CHAP. 13)
David McAllister North Carolina State University (CHAP. 14)
John Munson University of Idaho (CHAP. 12)
John Musa AT&T Bell Laboratories (CHAP. 5, 6)
Alien Nikora Jet Propulsion Laboratory (CHAP. 7)
George Stark Mitre Corporation (APP. A)
Chapter 17. Neural Networks for Software Reliability Engineering

17.1 Introduction
17.2 Neural Networks
   17.2.1 Processing Unit
   17.2.2 Architecture
   17.2.3 Learning Algorithms
   17.2.4 Back-Propagation Learning Algorithm
   17.2.5 Cascade-Correlation Learning Architecture
17.3 Application of Neural Networks for Software Reliability
   17.3.1 Dynamic Reliability Growth Modeling
   17.3.2 Identifying Fault-Prone Modules
17.4 Software Reliability Growth Modeling
   17.4.1 Training Regimes
   17.4.2 Data Representation Issue
   17.4.3 A Prediction Experiment
   17.4.4 Analysis of Neural Network Models
17.5 Identification of Fault-Prone Software Modules
   17.5.1 Identification of Fault-Prone Modules Using Software Metrics
   17.5.2 Data Set Used
   17.5.3 Classifiers Compared
   17.5.4 Data Representation
   17.5.5 Training Data Selection
   17.5.6 Experimental Approach
   17.5.7 Results
17.6 Summary Problems

Appendix A. Software Reliability Tools

Appendix B. Review of Reliability Theory, Analytical Techniques, and Basic Statistics

References  781
Index  821

Contributors

Robert Tausswothte Jet Propulsion Laboratory (CHAP. 16) MIaden
Vouk North Carolina State University (CHAP. 11, 14) Geoff Wilson
AT&T Bell Laboratories (CHAP. 6)
In complex software systems, reliability is the most important aspect of software quality, but one that has often been the most elusive to achieve. Since more and more of the world’s activities and systems are dependent on software, achieving the appropriate level of software reliability consistently and economically is crucial. Software failures make newspaper headlines because at best they inconvenience people and in extreme cases kill them.

It is refreshing to see a book that has the potential to make a significant improvement in software reliability. The *Handbook of Software Reliability Engineering* is an important milestone in the history of software reliability engineering. Michael R. Lyu has assembled a team of leading experts to document the best current practices in the field. The coverage is comprehensive, including material on fault prevention, fault removal, fault tolerance, and failure forecasting. Theory, models, metrics, measurements, processes, analysis, and estimation techniques are presented. The book is filled with proven methods, illustrative examples, and representative test results from working systems in the field. An important component of the book is a set of reliability tools that can be used to apply the techniques presented.

The subject is treated with the rigor that is characteristic of a mature engineering discipline. The book stresses mathematical models for evaluating reliability trade-offs, and shows how these models can be applied to the development of software systems.

With the publication of this Handbook, the field of software reliability engineering has come of age. This book is must reading for all software engineers concerned with software reliability.

*Alfred V. Aho*
Early in this exhaustive treatment of what may be the single most critical aspect of modern software development, the editor says "Mature engineering fields classify and organize proven solutions in handbooks so that most engineers can consistently handle complicated but routine designs." The reliability engineering of software has become mature with the appearance of this Handbook.

In my graduate software engineering course, I motivate the importance of early test planning with reliability requirement setting examples. It is, in my experience an issue about which success or failure of major systems projects revolve. In the early 1980s I led the DOD's software testing and reliability analysis team for the final operational tests of the now-famous Patriot Missile System. The questions? What was the required system reliability? Was the operational test data consistent with these requirements? Not many people know how close Patriot came to being rejected as a viable weapons system—not because the system itself was bad, but rather because the reliability engineering was so flawed that developers could no determine how reliable it really was. This crisis could have been avoided had software reliability engineering practice been systematized and applied in the manner advocated by this Handbook.

Reliability theory and engineering statistics textbooks ignore software, for the most part. Software engineering textbooks generally ignore reliability theory. Classroom teachers of the subject are forced to the kind of anecdotal material mentioned above, perhaps augmented by special-purpose supplementary readings. Even worse, software reliability theory has a reputation for facileness that has been encouraged by the many contributors who try to apply hardware reliability models mutatis mutandis to the very different (and more difficult) problems of software reliability.

So, when I was asked by the editor to review this Handbook, I agreed eagerly. On the one hand, a "real" handbook would be of inestimable
help to practitioners, decision makers, teachers, and students. On the other hand, a spotty or imbalanced treatment would only make matters worse. I said I would offer my comments only after reading the entire book.

The first thing I did when I received the manuscript was to check it against my classroom "staples." There for the first time in book form was a coherent approach to developing reliability requirements. There also was a discussion of the relationship between software test and reliability estimation, the impact of software architecture on reliability, error studies and software fault classification, tools and methods extracted from best-practice benchmarks of the best reliability labs in the world, actual data. It was all there—and in pretty much the same form in which I would have presented it myself. The editor even included exercises to make it suitable for classroom use.

Encouraged, I read the manuscript front to back. This is a book that will be the standard by which the field is measured for years to come. It is thorough, correct, readable, and so current that it actually anticipates results that have not appeared in archival journals yet. It contains the best work of many of the founders of the field. It contains innovations by some of the rising stars. It is, however, more than anything else a Handbook in the tradition of the classic handbooks of mathematics, physics, and engineering. It does not present software reliability as a silver bullet. It does not attempt to proscribe the complex system usages that would require skill and training on the part of software developers. Rather it seeks to "... classify and organize proven solutions ... so that most engineers can consistently handle complicated but routine designs." In this it succeeds, far beyond my expectations. It clearly establishes software reliability engineering as a mature engineering discipline.

Richard A. DeMillo
Ever since I entered the field of software reliability engineering some years ago, I have been looking for a book that exclusively and comprehensively deals with software reliability subjects that interest me, as both a researcher and a practitioner. I wasn’t able to find one. So I started this project by inviting the leading experts in this field to contribute chapters for this book. I laid out the framework of the book, identified its essential components, and integrated them by maintaining completeness and avoiding redundancies. As an editor, my duty is to ensure breadth, while the chapter authors treat the subjects of their delegated chapters in depth.

This is a handbook on software reliability engineering. The theme underlying the book is the formulation, application, and evaluation of software reliability engineering techniques in practice. Reliability is obviously related to many characteristics of the software product and development process. This Handbook intends to address all its aspects in a quantitative way.

The book is designed for practitioners or researchers at all levels of competency, from novice to expert. It is targeted for several large, general groups of people who need information on software reliability engineering. They include:

1. People who need a general understanding of software reliability. These are high-level managers, professional engineers who use software or whose designs interface with software, and people who acquire, purchase, lease, or use software.
2. Software developers, testers, and quality assurance personnel who use and apply software reliability engineering techniques. This also includes practitioners in related disciplines such as system engineering, reliability management, risk analysis, management-decision sciences, and software maintenance.
3. Researchers and students in software engineering, reliability analysis, applied statistics, operations research, and related disciplines, and anyone who wants a deeper understanding of software reliability and its engineering techniques.

Each of the book’s individual topics (i.e., chapters) could be considered as a compact self-contained minibook. However, these topics are presented in relation to the basic principles and practices of software reliability engineering. The approach is to provide framework and a set of techniques for evaluating and improving the engineering of software reliability. It presents specific solutions, obtained mostly from real-world projects and experimental studies, for routine applications. It further highlights promising emerging techniques for research and exploration opportunities.

The book has been thoroughly indexed for your convenience, so that it can serve as a true handbook, and a comprehensive list of references is provided for the purpose of literature search. As a unique value-added feature, this book includes a CD-ROM, which contains 40 published and unpublished software project failure data sets and some of the most advanced software reliability tools for ready application of software reliability techniques and a jump-start on software reliability engineering programs.

This book is also designed to be used as a textbook by students of software engineering or system reliability, either in a classroom or for self-study. Examples, case studies, and problems have been provided throughout the book to illustrate the concepts and to walk through the techniques. A Solution Manual is available from the editor with solutions to some of the exercises.

What is finally presented here is the work of celebrated international experts contributing their most advanced knowledge and practices on specific reliability-related topics. The development team of this book wants to thank our colleagues who provided continuous encouragement and thorough review of the chapters of the book. They are Jean Arlat, Phillip Babcock, Farokh B. Bastani, Brian Beckman, Justin Biddle, James Bieman, Harry S. Burns, Sid Dalal, Chris Dale, Adrian Dolinsky, George Finelli, Amrit Goel, Jack Goldberg, Myron Hecht, Walter Heimerding, Yu-Yun Ho, Yennun Huang, Robert Jackson, Mohamed Kaaniche, Kalai Kalaichelvan, Rick Karcich, Ted Keller, Elaine Keramidas, Chandra Kintala, Sy-Yen Kuo, Ming Y. Lai, Alice Lee, Haim Levendel, Yi-Bing Lin, Peng Lu, Richard E. Machol, Suku Nair, Mits Ohba, Gardner Patton, Hoang Pham, Francesca Saglietti, Norm Schneidewind, Robert Sherman, David Siegel, Pradip Srimani, Mark Sullivan, Robert Swarc, K.C. Tai, Yoshi Tohma, Randy Van Buren, C.W. Vowell, Anneliese von Mayrhauser, Chris J. Walter, Yi-Ming Wang, Pramod Warty, Chuck Weinstock, Min Xie, and Jinsong Yu.
We are most appreciative of the organizations and projects that provided funding for the work conducted in some of the book chapters. They are the Advanced Research Projects Agency, the ESPRIT Basic Research Action on Predictably Dependable Computing Systems, the ESPRIT programme as part of the PDCS1 and PDCS2 projects, the EU Environment programme as part of the SHIP project, IBM at Pough-keepseie, New York, the Illinois Computer Laboratory for Aerospace Systems and Software (ICLASS), National Aeronautics and Space Administration (NASA), NASAAMES Research Center, Office of Naval Research, Tandem Computers Incorporated, the U.K. EPSRC as part of the DATUM project, and the U.S. Air Force Operational Test and Evaluation Center (AFOTEC).

I also want to particularly thank Al Aho and Rich DeMillo for writing forewords to this book. Their comments are helpful and rewarding. I am greatly thankful to Karen Newcomb of NASA COSMIC and LiHam Valdez-Diaz of AT&T for permission to include CASRE, SoftRel, and AT&T SRE Toolkit in this book. My appreciation goes to Jean Glasser, Marjorie Spencer, John Wyzaiek, and Suzanne Rapcavage, editing and production supervisors at McGraw-Hill during different stages of this book. Midge Haramis’s assistance is also acknowledged. The invaluable guidance and help of Christine Furry at North Market Street Graphics during many revision, editing, and production cycles have also made this book project much easier than it would have been.

Finally, I want to thank my wife Felicia, to whom this book is dedicated.

Michael R. Lyu Murray Hill, New Jersey